UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

TESIS DOCTORAL

DESCUBRIMIENTO DINÁMICO DE SERVIDORES BASADO EN INFORMACIÓN DE LOCALIZACIÓN USANDO UNA TABLA DE HASH DISTRIBUIDA BALANCEADA

Autor: **Rubén Cuevas Rumín**
Ingeniero en Telecomunicación, Máster en Planificación y Gestión de Redes,
Máster en Ingeniería Telemática

Directora: **Carmen Guerrero López**
Ingeniera en Telecomunicación, Doctora en Ingeniería Informática

Leganés, Junio de 2010

UNIVERSIDAD CARLOS III DE MADRID

DEPARTMENT OF TELEMATICS ENGINEERING

Ph.D. THESIS

DYNAMIC AND LOCATION AWARE SERVER DISCOVERY BASED ON A FAIR DISTRIBUTED HASH TABLE

Author: **Rubén Cuevas Rumín, M.Sc.**

Supervisor: **Carmen Guerrero López, Ph.D.**

Leganés, June 2010

LOCATION AWARE SERVER DISCOVERY USING A FAIR DISTRIBUTED HASH TABLE

Autor: Rubén Cuevas Rumín

Director: Dra. Carmen Guerrero López

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Carlos III de Madrid, el día ___ de _____ de _____.

Firma del tribunal calificador:

Firma:

Presidente:

Vocal:

Vocal:

Vocal:

Secretario:

Calificación:

Leganés, ___ de _____ de _____.

A mis padres Loli y Ángel,
a mis hermanos Arantxa y Ángel,
a mi chica María,
a mi yaya Chon,
y a mi sobrino Celso.

El éxito consiste en alcanzar lo que se desea,
la felicidad en desear lo que se alcanza.

– Anónimo

Debe ser simple para ser cierto.
Si no es simple, probablemente no podremos descifrarlo.

– Albert Einstein (1879-1955)

Agradecimientos

Esta Tesis es la culminación de un proceso de formación largo, a veces duro pero muy satisfactorio que no podría haber sido posible sin una serie de personas que han estado a mi lado. De hecho aunque el nombre que aparecerá en el futuro título de Doctor será el mío hay muchos otros que deberían aparecer a mi lado pues son tan merecedores de ello como yo.

En primer lugar quiero agradecer a mis padres, Loli y Ángel, *todo*. Creo que es la palabra que mejor define lo que siempre han hecho por mi y lo que es estarían dispuestos a dar por mi. Se que aunque viviese cien vidas jamás podría devolveros todo lo que me habéis dado y enseñado. Soy lo que soy y como soy por vosotros, y este éxito en mi vida es tan vuestro como mío.

Quiero acordarme de mi hermano Ángel, mi mejor amigo, mi compañero de viaje, con el que comparto todo, la persona que mejor me conoce y la que más me ha ayudado. Simplemente gracias por ser como eres y por estar siempre conmigo (incluso cuando no estás).

También tengo que agradecer a mi hermana Arantxa, por ser un ejemplo y por su forma de ser y pensar que siempre me han ayudado a poner los pies en el suelo, por lo fácil que es arrancarte esa sonrisa que tanto me gusta. Por todo eso y muchas más cosas gracias.

Ahora te toca a ti María, tú eres la que cada día me regalas tu presencia para saber que soy una persona afortunada por poder estar a tu lado. Gracias por hacerme día a día feliz, por comprenderme, por aguantarme y apoyarme. Y lo más importante, gracias por ofrecerme un futuro lleno de sueños e ilusiones que vamos a compartir juntos.

La siguiente en la lista eres tú Yaya. Tú no sabes de estas cosas a las que yo me dedico, pero gran parte de mi éxito te lo debo a ti. Tú me has demostrado lo que significan cosas tan importantes como la lealtad, la fidelidad, la bondad, la fortaleza, el cariño desinteresado. Sí, porque tú eres la persona más leal, fiel, buena, fuerte y que ofrece el cariño más disenteresado que conozco. Todo eso lo he aprendido de ti, y todo eso me ha ayudado a alcanzar siempre mis objetivos en la vida de una manera humilde y honesta. Siempre te llevo en mi corazón.

Y ahora el pequeñajo, Celso. No llevas ni un año con nosotros y parece que llevas toda una vida. En este tiempo de mucha tensión y trabajo, tu eras mi mejor válvula de escape. Eres especial y como tú madre me ayudas a darme cuenta de las cosas que son realmente

importantes.

Quiero acordarme del resto de mi familia, mis abuelos Pilar y Lumi, mis tíos Javi, Fali, Loli, Pili, Carlos, mis primos, mi cuñado Celso y de mi otra familia Macu, Eugenio, Sara y Sole. Todos en mayor o menor medida han aportado su granito de arena para que hoy yo pueda estar convirtiendome en Doctor. En especial quiero acordarme de mi tía Mari que una vez junto con Isabel me apoyaron y me ayudaron. Espero haberles demostrado que valió la pena esa ayuda.

También quiero mencionar a mis *chavales* Rafa, Toñín, Victor, Diego, Alfonso, Jorge, Javichu, Antonio, Jose, Carlos y algún otro del que seguro me olvido. Gracias por tan grandes momentos y recuerdos. Espero que nos queden muchos más por vivir juntos.

Por último, en el aspecto personal, quiero acordarme de Soco, bueno mejor dicho de mi *tita* Soco. Se que lo estás pasando un poco mal, pero ya sabes que estamos contigo y esta es la mejor manera que se me ocurre de decírtelo. Verás como podemos con ello.

En el aspecto profesional quiero empezar agradeciendo a mi tutora Carmen, por estar siempre dispuesta a ayudarme, por lo fácil que ha sido trabajar contigo, por tus consejos y por el grado de complicidad que hemos alcanzado con el paso del tiempo.

Quiero agradecer especialmente a Albert, Nikos, Manolo y Ángel. Son las personas de las que más he aprendido y que más me han ayudado a convertirme en el investigador que soy hoy en día. Espero seguir teniendo el placer de trabajar con vosotros en el futuro.

También quiero agradecer a toda la gente con la que he colaborado en los diferentes proyectos y artículos que he realizado hasta ahora, destacando especialmente a Albert Cabellos y a Sebastian Kaune.

A todos mis compañeros de departamento: Isaías, Pablo, Carlos Jesús, Richi, Iván, Antonio, Jose Alberto, Jaime, Marcelo, Paco, Ignacio, María, Alberto, David, Manolo, Albert, Isaac, Mika, Marco, Chema, Andrés, Goyo y alguno que seguro se me olvida. Gracias.

Por último quiero agradecer a Arturo por haberme dado la oportunidad de dedicarme a lo que me gusta, y por ser una de las personas que más apuesta por una investigación de calidad en nuestro campo, una labor ardua, de gran importancia y no siempre reconocida.

Abstract

The current Internet includes a large number of distributed services. In order to guarantee the QoS of the communications in these services, a client has to select a close-by server with enough available resources. To achieve this objective, in this Thesis, we propose a simple and practical solution for Dynamic and Location Aware Server Discovery based on a Distributed Hash Table (DHT). Specifically, we decide to use a Chord DHT system (although any other DHT scheme can be used). In more detail, the solution works as follows. The servers offering a given service S form a Chord-like DHT. In addition, they register their location (topological and/or geographical) information in the DHT. Each client using the service S is connected to at least one server from the DHT. Eventually, a given client C realizes that it is connected to a server providing a bad QoS, then, it queries the DHT in order to find an appropriate server (i.e. a close-by server with enough available resources). We define 11 design criteria, and compare our solution to the Related Work based on them. We show that our solution is the most complete one. Furthermore, we validate the performance of our solution in two different scenarios: (i) NAT Traversal Server Discovery and (ii) Home Agent Discovery in Mobile IP scenarios. The former serves to validate our solution in a highly dynamic environment whereas the latter demonstrates the appropriateness of our solution in more classical environments where the servers are typically *always-on* hosts.

The extra overhead suffered from the servers involved in our system comes from their participation in the Chord DHT. Therefore, it is critical to fairly balance the load among all the servers. In our system as well as in other P2P systems (e.g. P2PSIP) the stored objects are small, then routing dominates the cost of publishing and retrieving objects. Therefore, in the second part of this Thesis, we address the issue of fairly balancing the routing load in Chord DHTs. We present an analytical model to evaluate the routing fairness of Chord based on the well accepted Jain's Fairness Index (FI). Our model shows that Chord performs poorly. Following this observation, we propose a simple enhancement to the Chord finger selection algorithm with the goal of mitigating this effect. The key advantage of our proposal as compared to previous approaches is that it adds a neglible overhead to the basic Chord algorithm. We validate the goodness of the proposed solution analytically and by large scale simulations.

Keywords: Location Aware, Server Discovery, DHT, Fairness, Chord.

Resumen

En los últimos años un gran número de servicios distribuídos han aparecido en Internet. Para garantizar la Calidad de Servicio de las comunicaciones en estos servicios sus clientes deben conectarse a un servidor cercano con suficientes recursos disponibles. Para alcanzar este objectivo, en esta Tesis, se propone una solución simple y práctica para el Descubrimiento Dinámico de Servidores basado en Información de Localización usando una Tabla de Hash Distribuída (DHT). En concreto, hemos decidido usar una DHT de tipo Chord (aunque cualquier otro tipo de DHT puede usarse). A continuación describimos brevemente nuestra solución. Los servidores que ofrecen un servicio específico S forman una DHT tipo Chord donde registran su información de localización (topológica y/o geográfica). Cada cliente que usa el servicio S está conectado al menos a un servidor de la DHT. En caso de que un cliente C perciba que el servidor al que está conectado está ofreciendo una mala Calidad de Servicio, C consulta la DHT para encontrar un servidor más apropiado (p.ej. un servidor cercano con suficientes recursos disponibles). En la Tesis se definen 11 criterios de diseño y se compara nuestra solución con las soluciones existentes en base a ellos, demostrando que la nuestra es la solución más completa. Además, validamos el rendimiento de nuestra solución en dos escenarios diferentes: (i) Descubrimiento de Servidores para atravesar Traductores de Direcciones de Red (NATs) y (ii) Descubrimiento de Agentes Hogar (HAs) en escenarios de Movilidad IP. El primero sirve para validar el rendimiento de nuestra solución en escenarios altamente dinámicos mientras que el segundo demuestra la validez de la solución en un escenario más clásico donde los servidores son máquinas que están ininterrumpidamente funcionando.

Los servidores involucrados en nuestro sistema sufren una sobrecarga debido a su participación en la DHT tipo Chord. Desafortunadamente, esta sobrecarga es inherente al sistema anteriormente descrito y no se puede eliminar. En cambio lo que sí podemos hacer es balancear la carga de la manera más justa posible entre todos los servidores. En nuestro sistema, al igual que en otros sistemas P2P (p.ej. P2PSIP) los objetos almacenados tienen un tamaño pequeño, produciendo que sea la tarea de enrutamiento la que domina el coste de publicar y obtener objetos. Por lo tanto, en la segunda parte de esta Tesis abordamos el reparto equilibrado de la carga de enrutamiento en DHTs tipo Chord. En primer lugar, definimos un modelo analítico para evaluar el reparto de la carga de enrutamiento entre los nodos que forman una DHT tipo Chord. Para ello nos basamos en una métrica

aceptada por la comunidad investigadora como es el Jain's Fairness Index (FI). El modelo resultante demuestra que Chord tiene un rendimiento pobre en el reparto justo de la carga de enrutamiento. Basándonos en esta observación proponemos una modificación simple al algoritmo de selección de punteros de Chord para mejorar el reparto de la carga de enrutamiento. La ventaja fundamental de nuestra solución en comparación con otras propuestas anteriores es que nuestra solución añade un coste despreciable al algoritmo básico de Chord. Finalmente, validamos el rendimiento de nuestra solución analíticamente y por medio de simulaciones a gran escala.

Palabras Clave: Localización, Descubrimiento de Servidores, DHT, Balanceo de Carga, Chord.

Contents

List of Figures

List of Tables

Chapter 1

Motivation and Introduction

In the last years the Internet services have evolved in an exponential fashion. The old client/server paradigm is rapidly moving to new schemes: (i) the servers are anymore single central machines but distributed architectures formed by hundred or even thousand of machines (e.g. CDNs, Distributed Data Centers), (ii) some applications rely in the new peer-to-peer (P2P) paradigm where the users play at the same time the role of client and server (e.g. file-sharing, Skype, Online Games), (iii) most of the current Internet services are large scale worldwide services.

In this new picture, there are worldwide deployed services where the client has to use a topologically/geographically close server in order to fulfil the service's QoS requirements. For instance, multimedia services must keep the communication delay under a given threshold (end-to-end VoIP communication delay must be lower than 150ms following ITU-T recommendation [itu96]). Furthermore, clients of other applications such as P2P or on-line gamming can benefit from the selection of close file sources and servers respectively.

The first objective of this Thesis is to design a solution whose main requirement is *selecting a close-by server that guarantees the QoS requirements of the communication*. For instance, in the case of VoIP applications the selected server must guarantee a end-to-end delay inferior to 150 ms based on the ITU-T recommendation [itu96] or 200 ms based on the Cisco recommendation [CIS05].

The problem of selecting a close-by server has been previously studied in the literature. IP Anycast was proposed as a network-layer solution to server selection. However, due to various deployment and scalability problems [KW00] it has not been deployed. Therefore, other proposals have addressed this problem using solutions based on overlay networks [WSS05, FLM06]. In addition, some other solutions that were not initially proposed to find a close-by server, can be adapted to solve this problem [MIP+06, FJJ+01, KR00, MIP+06, DCKM04, LGS07, AL09]. Unfortunately, these solutions present some problems that we would like to mitigate or eliminate.

Moreover, the selection of a close-by server is extremely important to P2P applications

(e.g. selecting a close source of a file among those available in a file-sharing system). The Internet Service Providers (ISPs) have recently shown their worry regarding the transit traffic generated by the P2P applications that is not under their control. A client of a P2P application typically selects its peers at random. This produces that the neighbourhood of a client is mainly composed by nodes belonging to different ISPs even if there are local available peers. The traffic exchanged with these remote neighbours goes to the transit and peering links of the ISP increasing its costs. Some ISPs have started to filter the traffic of some P2P applications such as BitTorrent [Coh03] in order to mitigate this problem. On the other side, the developers of P2P clients have started to implement techniques to avoid these filters [DMHG08] . To bring peace to this conflict, the research community has proposed solutions to keep as much traffic as possible within the client's ISP. These are known as *locality* or *ISP friendly* solutions. Since our proposal is likely to be used by P2P applications it is desirable designing it as an ISP-friendly solution.

In a nutshell, our system must fulfil the following requirements:

1. Fully Distributed
2. Do not use end-host probing
3. Robust to Triangular Inequality Violation
4. Robust to Churn
5. Robust to User Mobility
6. Low Measurement Overhead
7. Low Maintenance Overhead
8. Simplicity
9. Server selection based on distance/delay and other parameters such as server capacity and availability
10. Offer an Anycast service (i.e. to be able to support close-by server discovery for multiple services at the same time)
11. ISP friendliness

To achieve our objective we propose a *Simple* and *Practical* solution for *Dynamic and Location Aware Server Discovery* based on an overlay network. In the proposed system the Servers of a given service S form a Distributed Hash Table (DHT) where they store their location information. This location information can be from different nature: (i) Internet topological information such as the Autonomous System (AS) number or the IP Prefix; (ii) Geographical information such as geopolitical information (country, region, city) or geographic coordinates (GPS or latitude/longitude). Therefore, we have a logical infrastructure where we can find (topological and/or geographical) close-by servers to a given client. This

can be achieved by comparing the location information of the client to the location information stored by the servers in the DHT.

In more detail, the solution works as follows: when a server is switched on, it joins the DHT using the standard mechanism [SMLN+03], and registers its location information in the DHT (e.g. its AS number, its Country, etc..). On the other hand, every client is connected to at least one of the servers in the DHT. Thus, if a client C is connected to a server S offering a bad performance to the communications (e.g. the server is far away or the server is overloaded) the former lunches the Server Discovery procedure. It sends a *find server query* to S that uses the public IP address of C to infer its location information (e.g. AS number, Country, etc). After that, S sends a query within the DHT to find other servers located close to C (e.g. in the same AS). S obtains as result a list of the IP address of all (or a subset of all) the servers matching the search criteria. S forwards this list to C. Finally, C selects one of the servers based on a preconfigured criterion. For instance, the client can measure the Round Trip Time (RTT) to the obtained servers and selects the one offering the lowest delay.

In order to validate our solution, we apply it in two cases of study: NAT Traversal Servers (or Relays) Discovery [CCCA+10] and Home Agents Discovery (Mobile IP) [CCA+09, CACDP+09, CGC+09, CCUnG07, CCG+07]. In the first case the proposed solution is named Peer to Peer NAT Traversal Architecture (P2P-NTA) , whereas in the second case the solution is called flexible Peer to Peer Home Agent Network (fP2P-HN). These systems demonstrate the simplicity and practicability of our solution in two very different scenarios. In the former, we demonstrate that using our solution for Relays discovery in NAT Traversal scenarios (fundamentally P2P applications) we achieve a performance similar to the direct communications while reducing the transit traffic of the ISP hosting the Relay node. Whereas, in the latter we demonstrate that our service is also valid in a more classical Client/Server scenario, where the servers are expected to be *always on*. Next we briefly describe both cases.

The P2P-NTA is designed for dynamically discovering NAT Traversal Servers. Due to the rapid consumption of IP addresses, the Network Address Translators (NATs) have been adopted as a de-facto standard by ISPs and companies. This solution allows multiplexing the usage of a single public IP address in order to gain access to the Internet from a large number of computers. NATs work well in client/server architectures, where the client is the one opening the connections. However, in the last decade the P2P applications have broken this paradigm and now any end host in the Internet using a P2P application is server and client at the same time, thus, it opens but also receive incoming connections. Unfortunately, NATs do not allow incoming connection. This is a serious problem, since nowadays P2P applications are used by hundred of millions of end-users and represent a major part of the Internet traffic [Ipo07]. Some NAT Traversal protocols such as TURN [RMM09] or STUN [RWHM03] have been designed to allow the users behind NATs (NATed users) to receive connections. These protocols rely on a third entity named NAT Traversal Server or Relay that typically has a public IP address. It is worth noting, that in some P2P applications those users having some specific features such as a public IP address adopt the role of Re-

lays. Then, the Relays (i.e. these specific users) are just available when the user is running the P2P application. For instance in Skype these users are named Super Nodes.
The Relay is used by the NATed client to find out the type of NAT it is behind of. Depending on the type of NAT, the user could or could not establish/receive a direct communication to an external host. In the worst case, the host would use the Relay to establish and receive its communications. Then it is desirable to utilize a Relay that guarantees the communication QoS (e.g. a bounded delay and enough bandwidth).
On the other hand, some P2P applications (e.g. BitTorrent) generate a large amount of traffic that is not under the control of ISPs and produces extra costs to them. To solve this problem new *locality* solutions have been recently proposed. The rationale behind these solutions is to keep the P2P traffic local within the ISP. However, there is not a solution considering this problem when Relays are used. In the case of relayed communications, it is desirable that the Relay is located in the same ISP as one of the users involved in the communication. By doing so, the cost of the relayed communication is equivalent to the cost of the direct communication.
We have adapted our Dynamic Location Aware Server Discovery solution based on DHTs in order to select a Relay that: (i) reduces the relayed communication delay and (ii) reduces the transit traffic and in turn the cost of the ISPs where the Relays are located. For this purpose, the Relays upon joining the DHT store the information regarding its AS, its Country and its Continent. Thus, the clients of P2P applications look for: firstly, a Relay located in the same AS. If this does not exist, a Relay located in the same country. Finally, if there is not a Relay either in the same AS or in the same country, it looks for a Relay in the same Continent in order to avoid transcontinental links. We name to this simple algorithm, *the Gradual Proximity Algorithm* (GPA). It is worth noting that P2P applications such as Skype or BitTorrent have tens of millions of users concurrently on-line. Then, it is likely to find at least one Relay within the same AS. Therefore, considering that most of the P2P clients find a Relay in its own AS: (i) the communication delay is similar to that one of the direct communications (we add an intra AS delay that is typically low), (ii) the transit traffic cost of the ISPs where the Relays are located is dramatically decreased.
In order to evaluate the P2P-NTA we use the large scale dataset from the iPlane [iPl] project that gives delay measurements of millions of pair of Internet hosts. The iPlane data demonstrates that the delay grows as proposed by the GPA: intra-AS $<$ intra-country $<$ intra-continent $<$ inter-continent. Furthermore, we evaluate the relayed communication delay and the transit traffic cost in different scenarios using the proposed solution. We compare the results against the case of direct communications, random Relay selection and pre-established Relay selection. The results demonstrate that P2P-NTA performs similarly than direct communications when considering reasonable deployments. Only 5% of the relayed communications established through P2P-NTA experience and extra delay that may degrade the QoS. In addition, the extra transit traffic generated in the Relays' ISPs is just 6%.

The fP2P-HN is designed for mobility scenarios where mobile users (terminal or networks) use a third entity called Home Agent (HA) in order to establish their communications. In the standard Mobile IP protocols the Home Agent is located in the mobile node's administrative domain, and it is static and unique. Thus, when the mobile node is far from

its administrative domain (e.g. the mobile node is in Europe and belongs to an US domain) all the node communications must cross a transoceanic link. This is especially harmful when the mobile node wants to communicate with another node in Europe, in this case the communication has to unnecessarily cross a transoceanic link twice. This adds an important delay to the communication that may not be supported by certain applications such as multimedia (e.g. the maximum end-to-end delay defined by the ITU-T in VoIP communication is 150 ms).
There exist several solutions to mitigate the effects of this problem. On one hand, the Return Routability of Mobile IPv6 (MIPv6) [JPA04] allows the nodes to communicate directly after the communication establishment. Unfortunately, this cannot be applied to either Mobile IPv4 (MIPv4) [Per02] or Network Mobility (NEMO) [DWPT05]. A second family of solutions proposes to deploy several Home Agents across the world; thus the client can always use a close Home Agent avoiding high latency links. However, this solution needs a scalable method to dynamically discover a close HA among those spread across the world. For this purpose we have adapted our DHT-based solution to the problem of Dynamic and Location Aware Home Agents Discovery. The Home Agents form a DHT where they store their topological information (i.e. their Autonomous System Number). Each Mobile IP client is connected to an HA belonging to the DHT. Therefore, if the client and the HA are far away, the process to discover a close HA to the mobile client is launched by either the mobile client or the HA. Then, the HA sends a query within the DHT asking by the IP address of other Home Agents located in the same AS where the mobile client is currently located. The HA forwards the obtained HAs' IP addresses to the mobile client, that connects to one of those HAs (that are located within the same AS). Furthermore, we propose a mechanism to reduce the load supported by the Home Agents. On the other hand, since the fP2P-HN is a collaborative commercial service that may be run by ISPs, the security is a must for its success. Thus, we propose a security mechanism for the solution. We have also evaluated the performance of the proposed solution in terms of communication delay and the HA's supported load and compared it with the standard MIPv4 and NEMO solutions. The results show that the communication delay can be reduced from 23% (considering that just a minimum number of ISPs deploy the service) up to 80% (with a large deployment). Furthermore the load suffered from the HA is reduced from 54% in the worst case to nearly 100% (in the best case).

On the other hand, the proposed solution as well as its two presented versions (fP2P-HN and P2P-NTA) have been designed using a Chord-like DHT, although, they can be implemented using any kind of DHT system. We have decided to use Chord since it is one of the most studied DHT systems[1]. Furthermore it has been implemented as part of several deployed systems (e.g. [DKK+01,i3]) and has been selected as the mandatory DHT scheme to be used by the P2PSIP WG [p2p].

In our system the extra overhead supported by the servers is due to their participation in the DHT. Therefore, it is a critical issue for us to fairly balance the DHT load. In our solution as well as other DHT-based systems such as P2PSIP, the size of the stored objects is

[1]The original paper has more than 8k references according to Google Scholar.

relatively small (i.e. Servers Location Information), then routing dominates the bandwidth and latency costs of storing and finding an object [GS05]. Thus, in the second part of this Thesis we address the issue of balancing the routing load in Chord [CUnB09].

First, we analyse the routing fairness in standard Chord according to the Jain Fairness Index (a de-facto metric to evaluate the fairness) [JCH98]. The analysis concludes that Chord routing is unfair. The root cause is the id assignment used in Chord. As explained in Section 3.2 the node-ids and the object-keys are randomly placed in the Chord ring (because they are obtained from a hash operation). This makes some nodes being responsible of a larger portion of the id-space than others, this is having a larger *zone of responsibility*. Then, these nodes are likely to receive a larger number of inbound fingers. Since the number of routed messages is directly related to the number of inbound fingers, those nodes having a larger number of inbound fingers are likely to suffer from a higher routing overhead. In summary, the unfair distribution of the size of the zone of responsibility leads to an unfair distribution of the number of inbound fingers that in turn leads to an unfair distribution of the number of routed messages.

Based on our analysis, we propose a simple enhancement to the finger selection mechanism: the node randomly selects a finger among the default finger selected in Chord and its s successors. By doing so, a node having a large zone of responsibility shares the inbound fingers and the routing load that it would have in standard Chord with its s successors. We demonstrate analytically that this simple mechanism largely improves the routing fairness by increasing the JFI of Chord from $\sim$0.6 to $\sim$0.9. Furthermore, our mechanism adds a negligible overhead and does not incur any performance penalty.
Finally, we run large scale simulations, with up to 1M nodes, that validate our analysis and study several representative scenarios such as churn, Zipf content popularity distribution and heterogeneous nodes. The results demonstrate that our enhanced Chord (*e-Chord*) outperforms the standard Chord in all the studied scenarios.

In summary, the main contributions of this Thesis are:

- A novel and simple system for Dynamic Location Aware Server Discovery based on a DHT.
- The P2P-NTA: an adapted version of the proposed solution addressing the discovery of NAT Traversal Servers.
- The fP2P-HN: an adapted version of the proposed solution for dynamically discovering Home Agents in IP Mobility scenarios.
- The analysis of the routing fairness in Chord.
- A simple solution that largely improves the routing fairness in Chord.

The rest of the document is organized as follows. In Part I we present the Background and the Related Work of this Thesis. Specifically, Chapter 2 introduces the basic functionality of Chord-like DHTs whereas Chapter 3 summarizes the main existing solution for Location Aware Server Discovery as well as those solution addressing the load balancing

in Chord. We dedicate the Part II of this document to explain our proposed solution for Dynamic and Location Aware Server Discovery. We first introduce the problem in Chapter 4. Chapter 5 details the proposed solution and Chapter 6 compares our solution to the related work. Afterwards we present the cases of study, NAT Traversal Servers Discovery and Home Agents Discovery for Mobile IP scenarios, in Chapter 7 and 8 respectively. Part III and Chapter 9 are devoted to the study and improvement of the Fairness Routing in Chord. Finally, Part IV concludes the paper with: Conclusions (Chapter 10), Future Work (Chapter 11) and a list with the main contributions of this Thesis (Chapter 12).

Part I

Background and Related Work

Chapter 2

Background

A large number of overlay solutions have been developed in the last years in the Internet. In these solutions the nodes form an overlay network at the application layer on top of the existing physical network and perform typical network operations such as routing. This releases the overlay application from the restrictions imposed by the physical network and the operators. The Peer-to-Peer (P2P) applications are the most successful members within the family of overlay applications mainly pushed by file sharing applications such as BitTorrent and Emule, in addition to emerging applications such as P2PSIP [p2p]. Indeed, nowadays P2P applications are responsible for most of the Internet traffic [CAC07] . P2P systems are classified into unstructured P2P systems, such as Gnutella [KM02], and structured P2P systems or Distributed Hash Tables (DHTs), such as Chord [SMLN$^+$03], Kademlia [MM02] or Pastry [RD01]. Unstructured schemes do not have control over the data placement (typically each node store its own objects) and the nodes are connected to a set of random selected peers in the overlay. This structure makes finding an specific object costly, however it is robust to dynamic environments where the nodes freely join and leave the system (this phenomenon is known as *churn*). On the other hand, structured schemes have control over the data placement. The overlay network has a well defined topology (ring, tree, etc) where both the nodes and the objects have a fixed position. This minimizes the look-up cost, however making the overlay network robust to churn incurs a high management cost. When a node leaves the system, the rest of the nodes perform the needed operations to maintain the overlay network connected and to avoid losing the objects stored by the departing node.

In this Thesis we present a solution for Dynamic and Location Aware Server Discovery based on a DHT. Specifically, we decided to design our solution on top of a Chord-like DHT (although any other DHT scheme can be used). Furthermore, as second contribution of this Thesis we present a simple mechanism to improve the routing fairness in Chord. Therefore, in the rest of the section we devote our effort to detail Chord. Chord is one of the most important DHT schemes and it is being used in several applications deployed in the Internet (CFS [DKK$^+$01], i3 [i3] ,etc) . Furthermore it has been recently adopted by the P2PSIP WG [p2p] as the mandatory DHT-scheme to be implemented in P2PSIP architectures. We refer the reader interested in the functionality of other DHT schemes to [LCP$^+$05], moreover

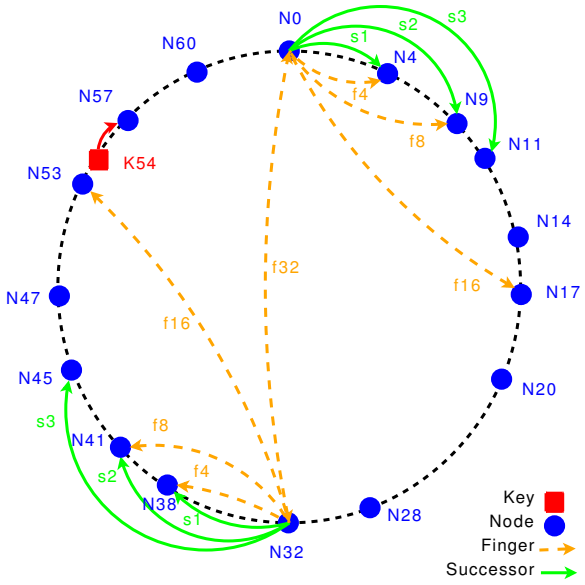


Figure 2.1: *An example Chord ring (k=6, s=3)*

for a more detailed description of Chord we recommend [SMLN+03].

A Chord P2P system is composed of nodes arranged in a logical ring (see figure 2.1). The position of each node in the ring is given by the node's identifier (id). Identifiers have a length of k bits which yields an identifier space $id \in [0, 2^k - 1]$. The objects stored in the ring are identified by their key, which belongs to the same identifier space as the node ids ($key \in [0, 2^k - 1]$). Nodes' identifiers and objects' keys are usually created by means of hash operations, which results in that nodes and keys are randomly placed into the Chord ring.

The responsibility for storing an object is given to the first node whose identifier is equal to or greater than the object's key, this is named key *successor*. Following this, we define the *zone of responsibility* of a node as the subset of the identifier space between this node and the previous one, since the node is responsible for all the keys that fall into this zone. For instance, in the example of figure 2.1, the object with key $K54$ will be stored by node $N57$, and its zone of responsibility is the id-space $(53, 57]$.

In Chord, to guarantee that the ring is not broken even when multiple nodes join and leave the system, each node knows the node with the immediate lower identifier to its own id and also the first s nodes whose identifier is greater than the node's. These nodes are referred to as the node's *predecessor* and *successors* respectively. Additionally, each node maintains a table of *fingers* which point to other nodes of the ring. In particular, a node with identifier id chooses as its fingers the nodes responsible for the keys $key_i = id + 2^{i-1}\ \forall i \in [1, k]$, respectively. In order to avoid duplicating information, the successors of a node are excluded from its finger table. Figure 2.1 shows the successors and fingers of nodes $N0$ and $N32$. Notice that $f1 = id + 2^0$, $f2 = id + 2^1$ are not shown because they point to the same node than $f4 = id + 2^2$. Notice also that nodes $N38$ and $N41$ are successors of node $N32$ and

therefore they are not included in its finger table.

Routing in Chord is based on the following key lookup algorithm. When a node receives a query looking for a given object's key, it looks into its table of fingers as well as its successors list for the nearest node to this key that has a lower or equal identifier, and forwards the query message to this node. In this way, the query message is routed through the Chord ring until it reaches a node that detects that the key is located between itself and its successor. This node then returns the IP address of the successor that is responsible for the requested object. In the example of figure 2.1, if node $N0$ looks for key $K54$, it would first use its finger $f32$ to query node $N32$. Then $N32$ would route the query message through its $f16$ finger, and finally node $N53$ would return a message to $N0$ with the IP address of node $N57$, which is the node responsible for key $K54$. A key feature of this lookup algorithm is that it provides a $O(log(n))$ performance, as each finger covers a zone of size 2^i and thus allows query messages to skip half of the identifier space left towards the key.

It is also worth noting that in order to assign replication capability to the Chord DHT, the object key can be stored in its $r \leq s$ successors, rather than the first successor.

In the rest of the paper we will use the following terminology for the Chord routing operation:

- We will say that a node sends a message over a *successor link* when it forwards it to one of its successors.
- With *outbound fingers* we will refer to the outgoing fingers of a node that point to other nodes. Following our previous explanation, the successors of a node will not be considered as part of its outbound fingers.
- With *inbound fingers* of a node we will refer to the incoming fingers from other nodes that point at this node. Similarly to the above, inbound fingers will not include those nodes that have the given node as a successor.

Chapter 3

Related Work

3.1 Related Work in Dynamic and Location Aware Service Discovery

The first part of this Thesis aims *to discover a server that is able to keep the required QoS of the node's communications (low delay, available resources and not overloaded)*. To achieve this, the primary objective is to find a topologically close server, i.e. a server offering a low delay. Some previous works [FLM06, WSS05] address this same problem. Furthermore, there are some other previous proposals that try to map a real delay graph of the Internet [MIP+06,iPl] or a virtual delay map of a set of nodes [DCKM04,LGS07,AL09]. These systems could be used as basis to identify a server offering a low delay.

J. Guyton et al. presented an early work on static and centralized location of nearby servers of a distributed service [GS95]. The solution combined traceroutes and hopcount measurements to determine the closest replica. One year latter, R. Carter et al. [CC97] demonstrated that dynamic server selection is more efficient than static server selection due to the variability of route latency over time and the large divergence between hop count and latency. In parallel IP Anycast was proposed as a network-layer solution to server selection. It was first proposed in 1993 by the IETF RFC 1546. However, due to various deployment and scalability problems [KW00] it has not been widely deployed.

Most recent works have addressed the general problem of estimate the delay between nodes in large-scale networks. We divide the solutions into three families: *Graph Representation*, *Network Coordinate Systems* and *On-Demand Probing Solutions*. Although not all these solutions address the specific problem of close-by server discovery, all of them are relevant to the problem. We will explicitly indicate those solutions that address the close-by server discovery problem. Next, we present the most representative solutions of each family.

3.1.1 Graph Representation

In this family, the solutions aim to represent a graph where the vertexes are the nodes participating in the system (e.g. Internet) and the edge between two nodes represents the distance (e.g. delay) between them. All-to-all probing solutions, as RON [ABKM01], produce accurate latency predictions but suffer from poor scalability. In order to achieve a higher scalability other proposals use clustering techniques to create the graph. Examples of these solutions are IDMaps [FJJ+01], Dynamic Distance Map [KR00] and the most recent and ambitious iPlane project [MIP+06, iPl]. Furthermore, it is worth noting that these solutions use a distributed infrastructure formed by Vantage Points in order to perform the measurements. Next we describe the different approaches.

3.1.1.1 IDmaps

IDMaps aims to create a graph of the Internet that allows estimating the distance between any pair of arbitrary nodes. For this purpose they use a set of Vantage Points distributed across Internet, named *Tracers*. Furthermore, they identify globally reachable Address Prefixes (AP). An AP is defined as a consecutive address range of IP addresses within which all hosts with assigned addresses are equidistant (with some tolerance) to the rest of the Internet. Therefore, given a set of Tracers and a set of APs, the distances between the Tracers as well as the distances between APs and their nearest Tracer(s) are measured. Thus, the distance between any two APs is the sum of the distance from each AP to its nearest Tracer and the distance between the two Tracers.

The number of used Tracers defines a trade-off between the system's scalability and accuracy. The more Tracers are used the more accurate are the measurements since a larger number of points of the Internet are covered and also the distance between the APs to the closest Tracer is smaller. On the other hand, IDMaps requires an all-to-all probing between the Tracers, thus the larger the number of Tracers is the higher the measurement load is.

Given the previous methodology, the key points affecting the accuracy of the system are: (i) Define the number and location of the Tracers and (ii) Identify the APs. In order to identify the number and location of the Tracers, the authors propose several graph theoretic approaches (k-HST and minimum K-Center) and heuristic (place the Tracers in Stub-ASes, place the Tracers in Transit-ASes and a mixed approach). Based on simulation results the authors claim that the minimum K-center algorithm and the Transit-AS heuristic are the best approaches.

Furthermore, the authors propose some improvements to the basic version of IDMaps. For instance, in order to avoid all-to-all probing the authors propose to identify nearby tracers and stop the probing between them.

3.1.1.2 Dynamic Distance Maps

This approach is based on a similar concept than IDMaps: (i) they use a set of Vantage Points named *mServers* in this case, (ii) they measure the distance between the mServers, (iii) they assign each end-host to the most closely connected mServer and (iv) they estimate the distance between two end-hosts as the distance between their two assigned mServers.

Therefore the main difference between IDMaps and this approach is that IDMaps estimates the distance between the Vantage Points and the end users, whereas this approach does not do it. Thus, the Dynamic Distance Map needs a larger set of Vantage Points in order to be accurate enough. This would lead to scalability problems. In order to reduce the measurement load, the mServers are grouped into clusters and the clusters organized in a hierarchical tree. A perfect clustering would require an all-to-all probing. To avoid this, a subset of mServers are randomly selected for creating an initial cluster structure. Afterwards, a Cluster Representative is selected for each cluster (this is the mServer in the cluster for which the maximum distance to other nodes within the cluster is minimal). The mServers that do not participate in the initial creation of clusters measure the distance to all the Cluster Representatives and join the cluster with the closest Cluster Representative. Finally, the distance between clusters is estimated as the average of several measured distances between members of the clusters. It is worth noting that the approach of clustering enforces symmetric distances. This is, the distance measured from the node to the cluster must be similar to the distance measured from the cluster to the node, otherwise the system cannot allocate the mServer in the appropriate cluster. Unfortunately, it is quite common to find asymmetries in the current Internet due to different reasons (e.g. routing policies). Furthermore, the proposed solution requires a periodical update of the Clusters Tree that produces an extra overhead.

3.1.1.3 iPlane

Most ambitiously than the previous approaches, iPlane seeks to create an *atlas* of the Internet that allows to characterize the end-to-end path properties (e.g. Delay, Loss Rate, Capacity, Available Bandwidth, etc) between any pair of nodes.

In order to achieve an accurate yet scalable Internet map, iPlane selects representative end-hosts to perform active probing on them. These representatives are selected as follows: firstly, they obtain a list of all globally routable prefixes in BGP, and selects an IP address within each prefix that responds to either UDP or ICMP probes. To improve the scalability and reduce the measurement load, they group the IP prefixes into *BGP atoms*. A BGP atom is composed by all the prefixes that have a common AS Path when measured from any given Vantage Point. Thus, a client's performance is approximated by a representative target in the same atom as the client. The process of BGP atoms identification is performed once every two weeks.

Planet Lab servers, located in more than 300 sites around the world, are used to perform exhaustive active probing of all the target representatives. Furthermore, Traceroute servers are configured to make low-intensity probing that helps to gather information regarding local

routing policies.

Traceroute probing gives as result a list of network interfaces on the path from source to destination. However, it is likely that interfaces on the same router, or in the same PoP may have a similar performance. Then, iPlane creates clusters of interfaces that allow to define a more compact and scalable topology that permits to perform more in-depth measurements and predictions. Several interfaces cluster techniques are used, for instance, interfaces belonging to the same PoP and interfaces within geographically nearby portions of the same AS are assigned to the same cluster. In a nutshell, iPlane use clustering to create a compact routing topology, where each *node* in the topology is a cluster of interfaces and each *link* connects two clusters.

iPlane authors claim that using this compact topology they are able to *measure* the Internet core. There is a centralized agent that orchestrates the measurement procedure by assigning to each Vantage Point the links it has to measure. For this purpose, the centralized agent balances the measurement load assigned to each Vantage Point. Furthermore in order to reduce the measurement load, only few Vantage Points measure each link. Finally, the centralized agent is also responsible of monitoring the execution of the tasks. With such a infrastructure iPlane uses a set of previously proposed techniques to estimate different link attributes:

- *Delay*: Latencies of the path segments can be derived directly from the traceroute data gathered while mapping the topology.
- *Loss Rate*: The Vantage points sends several large ICMP packets to the routers in the core. The estimated loss rate is obtained from the percentage of probes without responses.
- *Capacity*: iPlane uses previously defined methods [Bel92, Jac] that rely on a set of probe packets of different sizes to estimate the link capacity.
- *Available Bandwidth*: iPlane uses packet dispersion techniques [SKK03,HS03,JD03].

Unfortunately, most of these techniques are unlikely to work in end-hosts (e.g. home PCs). They rely on active probing that require the end-hosts to respond to unsolicited ICMP, UDP or TCP packets. Therefore, in order to gather measurement from end-hosts IPlane collects measurements from thousand of users participating in the BitTorrent application. On one side, they perform traceroutes to those end-hosts connecting to the iPlane measurement node. On the other side, they collect the packet traces of TCP flows to end-hosts in order to estimate loss rates, round trip times, etc.

With the described architecture iPlane predicts the path properties between an arbitrary pair of nodes in a two steps algorithm:

1. iPlane predicts the forward and reverse paths connecting the two nodes: if we consider a source S and a destination D, if S is a Vantage Point, iPlane simply returns the measured path from S to D. Else, iPlane looks for the closest intersection I of a

measured path between a Vantage Point V and D to S. Thus the path from S to D is estimated as S to I to D. Furthermore, the destination is typically represented by its BGP atom. Thus a more refined estimated path would be S to I to $D's\ atom$ to D[1].

2. iPlane aggregates the measured link-level properties to predict the end-to-end path properties: iPlane aggregates the measurements of the different segments forming the predicted path in order to obtain the end-to-end path performance. For instance the end-to-end RTT is estimated as the sum of the delay measured in the segments of the forward and the reverse path.

Finally, it is worth to note that iPlane data is publicly available through a replicated database.

3.1.1.4 Discussion

The Graph Representation Systems have been crucial to help the research community to further understand the behaviour of large scale networks such as Internet. Furthermore, they have been used as a tool for design and evaluate several large scale systems. However, in this Thesis we are interested in its applicability to the problem of nearby server discovery. Unfortunately, these solutions cannot be immediately applied to this problem. They could constitute a subpart (devoted to obtain the necessary location information) of a global system dedicated to the location aware server discovery. Even in this case the Graph Representation Systems have some drawbacks that reduce their attractiveness as solution to the problem of close-by server location:

- These systems need a measurement infrastructure of Vantage Points, thus there exist a trade-off between accuracy and scalability or measurement load. The more Vantage Points used the higher the measurement load, thus the poorer the scalability.
- They rely on heavy probing and measurement tasks that are costly. Furthermore, in some cases they need some central component able to orchestrate the measurement procedure (e.g. iPlane).
- They are sensitive to the high variance experienced by the path properties (delay, bandwidth, distance) in the current Internet. Therefore, the more robust they want to be to this variance the lower must be the periodicity of the measurements, thus a higher load they incur.
- They are sensitive to the Triangle Inequality Violation (TIV). Given a graph G, with a set of vertices V, a cost function $C(a, b)$ satisfies the triangle inequality if for all vertices a,b,c in G, C(a,c) $<$ C(a,b) + C(b,c). Then, if we consider this problem in the Graph Representation systems, we know the distance (delay) between a source node (s) and one Vantage Point (V). In the best case, we also knows the distance (or

[1] If there is a measurement of the last hop from D to its atom it is used, otherwise, a representative node in the atom is used (e.g. from BitTorrent measurements).

delay) between V and the destination node (d)[2]. If we consider this best scenario, the Graph Representation systems requires that the distance or delay function (d) satisfies the triangle inequality (d(s,d) $\geq$ d(s,V)+d(V,d)) in order to offer accurate results. Unfortunately, this assumption is continuously violated in the current Internet due to multiple reasons, e.g. policy-based routing.

3.1.2 Network Coordinate Systems

The Network Coordinate Systems (NCSs) embed the graph of the desired network (e.g. Internet) into a virtual coordinate space. Thus, the distance between two nodes in the virtual space is a representation of the distance/delay between them in the actual system. Therefore, the designers of these systems claim that the reduction in the representation has two main advantages in comparison to the Graph Representation Systems: (i) the amount of probing necessary to keep the data up to date is reduced and (ii) it enables decentralized implementations, this is, distributed NCSs where infrastructure measurement nodes (i.e. Vantage Points) are anymore needed.

In this subsection we briefly overview some of the most representative NCSs using infrastructure measurement nodes. However, the most relevant NCSs for us are those that are fully distributed. These proposals use graph embedding techniques that rely exclusively in measurement between the nodes. The nodes refine their position in the global coordinate space with their distance measurements to other nodes. We devote the major part of this subsection to detail three of the most representative distributed NCSs: Vivaldi [DCKM04], Pyxida [LGS07] and Htrae [AL09].

3.1.2.1 Non-distributed NCSs

The idea of NCS was first proposed for use in GNP [NZ01]. This systems use a set of *Landmarks* (i.e. Vantage Points) and obtains the RTT between each pair of Landmarks. With this information the system computes the virtual coordinates for each Landmark in such a way that minimizes the squared error in RTT estimation over all Landmarks pairs. The rest of the nodes measure their RTT to each landmark and, based on this each node configures its virtual coordinates to minimize the sum of the squared error in RTT estimation over all paths from itself to the landmarks. The main problem of this approach is that the landmarks become bottlenecks. Further approaches aim to alleviate this problem: Lighthouses [PCW$^+$03] uses multiple sets of landmarks using different coordinate spaces. A sophisticated method allows to unify the different spaces in such a way that the distance between coordinates in different spaces can be calculated. In PIC [CCRK03] each node that obtains its coordinates become a potential Landmark for new nodes.

[2]In the typical case, the distance between s and d is composed by the sum of the distance between several Vantage points located between s and d.

3.1.2.2 Vivaldi

The Vivaldi System calculates the network coordinates as the solution of a spring relaxation problem. In other words, Vivaldi minimizes a squared error function given complete knowledge of RTTs in the network.

In Vivaldi a node i as an associated coordinate x_i and its confidence in this coordinate is w_i (the confidence increases as it approaches 0). Furthermore, the node has some neighbours from/to whom it receives/sends RTT probes. The main contribution of Vivaldi is an algorithm to update the nodes coordinates based on the RTT measured between two nodes. Let's assume that node i has measured its RTT to node j, we denote it as rtt_{ij}. Furthermore, node i is assumed to know the following information: (i) both nodes coordinates (x_i and x_j) and coordinate confidence (w_i and w_j) and (ii) two preconfigured constants c_e and c_c that control the maximum impact that a single observation can have on the modification of the confidence and the coordinates, respectively.

With this information, the algorithm first calculates the *sample confidence* (w_s) and the *relative error* (ϵ). w_s represents the confidence the node must assign to this sample based on the nodes' confidence, whereas ϵ expresses the the accuracy of the coordinate (x_i) compared to the measured RTT. They are calculated as:

$$w_s = \frac{w_i}{w_i + w_j} \tag{3.1}$$

$$\epsilon = \frac{|||x_i - x_j|| - rtt_{ij}|}{rtt_{ij}} \tag{3.2}$$

Second, the algorithm updates the node's confidence (w_i) with an exponentially-weighted moving average (EWMA). The parameter α for the EWMA algorithm is set as $\alpha = c_e \times w_s$. Then, w_i is updated as:

$$w_i = (\alpha \times \epsilon) \times ((1 - \alpha) \times w_i) \tag{3.3}$$

Finally, the coordinate (x_i) is updated using the next expression:

$$x_i = x_i + \delta \times (||x_i - x_j|| - rtt_{ij}) \times u(x_i - x_j) \tag{3.4}$$

where δ is calculated as $\delta = c_c \times w_s$ and u indicates the direction of the coordinate correction.

The described algorithm is inspired by analogy to a real-world mass-spring system, thus it is designed for a three-dimensional Euclidean space. The authors discuss in the paper the possibility of using Euclidean spaces with more dimensions and conclude that the use of extra dimensions introduces a low gain in the accuracy of the system while adding a communication overhead. Furthermore, they also study the use of spherical coordinates and conclude that they do not model Internet well. Finally, the authors decide to apply two

dimension Euclidean space with *height vectors*. A height vector consists of a Euclidean coordinate augmented with a height. The Euclidean part models the latency in the Internet core that is proportional to the geographical distance whereas the height models the delay in the access link from the node to the Internet core. Hence, the distance between two nodes is the sum of the distance in the Euclidean space and the height associated to each node.

3.1.2.3 Pyxida

Pyxida is a distributed NCS based on Vivaldi. It is implemented in Azureus, one of the most used BitTorrent clients. Pyxida operates over the Azureus DHT in order to find nearby nodes for: (i) accelerating the overlay routing in the DHT and (ii) performing application-level congestion monitoring. The major contribution of Pyxida, apart from its technical improvements on Vivaldi algorithm, is that it is the first NCS implemented in a real large scale system.

The authors first analyse different properties of NCSs: Triangle Inequality Violation, Dimensionality and Type of Coordinate Space. For this purpose they use three latencies datasets: Azureus, PlanetLab and the MIT King dataset. First, they find that 83% of the Azureus node pairs, 85% of the MIT King and 68% of the PlanetLab violate the triangle inequality. Furthermore the authors measure a higher spread of the RTT in the Azureus dataset, they conjecture that this is a consequence of the traffic limitation imposed by some institutions and providers to the BitTorrent traffic [DMHG08]. Then, the authors conclude that the violations of Azureus to the triangle inequality produce a higher embedding error. Second, the authors study the appropriate dimensionality for Internet-scale network coordinates using the aforementioned datasets. They conclude that 4-5 dimensions is the most appropriate. Finally, in order to identify the most accurate coordinate space for Internet-scale networks, the authors study the intercontinental latency distribution. Interestingly they find that all the links between Asia and Europe crosses trough North America. This finding suggests that Internet topology is better represented by either an euclidean or an hyperbolic space than by spherical coordinates.

It is also worth nothing that Pyxida takes advantage of being embedded into a real system and it uses maintenance messages of the Azureus DHT to piggyback the needed information for the NCS. This reduces the probing overhead. On the downside, the nodes cannot select the neighbours to perform the delay measurements, thus affecting to the accuracy of the system.

From the technical point of view, Pyxida modify some aspects from the Vivaldi algorithm. Furthermore, it adds some mechanisms to mitigate the negative consequences derived from the piggybacking technique:

- *Latency Filter and Update Filter*: The former is used to make the coordinate to be more precise whereas the latter addresses the problem of making the coordinate stable across time. The latency measurements can be affected by two main effects: wrong measurements and underlay changes (network congestion or BGP routes changes). Pyxida authors find that a simple, short, moving median works well as

latency filter compensating both previous effects. On the other hand, the update filter makes a distinction between constantly evolving "system-level" coordinates and stable "application-level" coordinates. The former are used to refine the coordinate once the node is located in the correct area of the coordinate space, whereas, the latter indicates big changes in the underlay topology that makes the node to move to a different area of the coordinate space. Pyxida uses heuristics to compare windows of subsequent system-level coordinates in order to identify a migration of the node's application-level coordinates.

- *Neighbour Decay*: Due to the churn and the dependency in other application messages to piggyback the Pyxida data, the nodes do not have a fixed set of neighbours with which they could expect regular exchanges. The authors claim that in many cases the nodes just receive from 1 to 3 updates from a given node. Therefore if some nodes are sampled at a much higher rate than others, the local coordinate is skewed towards those nodes. In order to solve this problem, Pyxida does not run the coordinate update algorithm for each received sample; instead it runs the algorithm periodically considering all the information received during the last period. In the implementation only the nodes known in the last 30 minutes are used for the coordinate update. Furthermore the influence of each node is weighted by the time of last obtained sample and the number of samples associated to that node.

- *Neighbour Error*: Pyxida does not calculate the relative error (see ϵ definition in Vivaldi) of each single measurement. The authors claim that doing it per each sample: (i) may produce a large influence of individual wrong measurements and (ii) may become skewed due to the utilization of a reduced working set of nodes (because of the passive gossiping technique). Rather, Pyxida calculates the neighbour error, this is the distribution of relative errors for a set of recently contacted nodes.

- *4D+H Coordinates*: Pyxida uses a four dimension Euclidean space with height vector, rather than the 2D+H space proposed by Vivaldi.

3.1.2.4 Htrae

Htrae is a distributed NCS used for *matchmaking* in Online Games. In Online Gamming the user experience is directly affected by the lag in the direct communications between users. In order to provide a satisfactory experience the system must perform an accurate matchmaking, this is selecting groups of players with low latency between them.

One of the major contributions of the paper is the evaluation using a very large dataset composed by traces of millions of Xbox 360 consoles participating in matchmaking for the popular game Halo 3.

In summary, Htrae is a NCS using: (i) Vivaldi coordinate update algorithm, (ii) Spherical coordinates and, (iii) Geographic Bootstrapping. Next, we detail these and other technical aspects from Htrae:

- *Spherical Coordinates*: Htrae is the first NCSs that successfully uses Spherical Coordinates. It adapts the coordinates update algorithm of Vivaldi to the spherical coordinates.

- *Geographical bootstrapping:* In Htrae all the nodes are bootstrapped in the system with a coordinate obtained from its longitude and latitude. Previous NCSs were affected by local minimum due to the utilization of inappropriate initial conditions. Htrae authors claim that the geographical bootstrapping assigns to each node an initial coordinate close to its optimum position thus avoiding the problem of local minimum. Furthermore, the authors claim that the success of Htrae in the usage of a Spherical Coordinate space is due to the geographic bootstrapping of all the nodes in the system.

- *History Prioritization*: As we have already mentioned TIV is an important problem of NCSs. Htrae mitigates it by using real measured RTT instead of coordinates distance when it is possible: every time a node learns the RTT to another, it stores it in its *history*. When a distance prediction to a given node is needed, if there is *history* records of that node, the most recently measured RTT is used instead of the RTT predicted by the coordinates. This technique has a main problem: it is sensitive to delay variations (e.g. change in underlay network conditions or change in routing policy). Furthermore, the authors do not discuss how frequently this technique can be successfully used.

- *Symmetric Updates*: The author propose, rather than other NCSs, that when node i measures the RTT to node j in order to update its coordinate also informs j about the measured RTT, thus also j can update its coordinate. This can be viewed as a reduction of the probing overhead since one measurement is used by two nodes. However, it requires an extra message to inform the other node about the measured RTT.

- *Implementation*: In the current implementation of Halo 3 game, the matchmaking involves selecting one node as server, and then choosing up to 16 other nodes to be clients. When a console joins the systems it decides to be either a server or a client and notifies it to a central service (Xbox LIVE matchmaking service). In return the client obtains a list of potential servers, and measures the delay to them by using a ping-based probing. Finally, it chooses the closest one. Unfortunately, the authors do not discuss how to integrate Htrae in the described matchmaking system. In other words, the paper misses a complete explanation of how the new matchmaking service would be implemented in the real world. A possibility would be that the coordinates of all the users acting as server were registered in the central Xbox LIVE matchmaking service. Thus when a new client s its coordinates to the matchmaking service it obtains in return the closest server.

Finally, in the evaluation part the authors analyse the problem of finding a server providing a delay below a given threshold. This is quite similar to the problem addressed by this Thesis. Surprisingly, the results obtained by using exclusively Geolocation are similar to those obtained by Htrae. This make us to raise the following question: *Is it necessary a complex measurement system (e.g. NCS or Graph Representation) to solve the problem of close-by server selection?*.

3.1.2.5 Discussion

The NCSs are good candidates to be applied to the problem of close-by server discovery. However, as occur with the Graph Representation solutions they must be embedded as part of a global system dedicated to Location Aware Server discovery. In the case of Htrae the authors claim that their system is used for that purpose, however they never describe the complete system in the paper. Following we summarize the main drawbacks that NCSs have to be applied in our problem:

- They incur an important measurement load. An exception on this is Pyxida, however we have seen that piggybacking the info of the NCS in other application messages produces lack of accuracy and bias in the coordinate of the nodes.

- The NCSs are highly sensitive to the Triangular Inequality Violation. Unfortunately, this problem is quite common in the current Internet. This may lead to an important number of wrong nodes' coordinates.

- The NCSs aim to obtain an stable estimation of the Internet graph. However, Internet paths suffer from a high variability, thus the estimation has to be frequently updated. Therefore, there is a trade-off between the measurement load and the *up to date* state of the system. The more updated the system is the higher the measurement load, on the other side the lower the measurement load the more sensitive the system is to underlay network changes.

- It is not clear how these solutions can deal with churn and mobility. Pyxida and Htrae propose that when a user leaves the system and reconnects afterwards, it uses the coordinates from the previous session. This seems to work for the scenario presented by Htrae where the node is a console that is unlikely to be moved. However, in a more general scenario the application can be installed in different terminals that are by definition mobile terminals (PDAs, mobile phones, laptops, etc). In this case, there are two situations to be considered: (i) a user that disconnects and connects with a different IP address in a different geographical location. Then, using the coordinate from the previous session would be a mistake. (ii) mobile users, that change their locations while moving (e.g. 3G mobile phones). It is not clear, how NCSs can address these issues.

3.1.3 On-Demand Probing Solutions

These solutions aim to solve the problem of close-by server selection, and present a complete system addressing it. Therefore, they do not try to obtain any kind of Internet estimation. Thus, we can confirm that they are the closest solutions to our problem. In this subsection we present the two most important solutions within this family: Meridian [WSS05] and OASIS [FLM06].

3.1.3.1 Meridian

Meridian is a framework for performing node selection based on network location. The authors describe how Meridian can be use in order to solve three common network location problems: (i) discovering the closest node to a targeted reference point, (ii) find a node that offers minimal latencies to a given set of nodes and (iii) finding a set of nodes in a region whose boundaries are defined by latency constrain. The first one is the relevant one for us: a given client tries to find the closest server to itself (i.e. the reference point).

The nodes in Meridian are organized in a loosely connected overlay network. Each node keeps track of a fix number of other nodes in the system, and organizes its neighbours into concentric non-overlapping rings. The ith ring is defined by its inner radius $r_i = \alpha s^{i-1}$ and its outer radius $R_i = \alpha s^i$. The first ring is delimited by 0 and α, and there is a maximum number of rings defined by i^*, thus all the rings whose $i > i^*$ are collapsed into a single ring that spans the range $[\alpha s^{i*}, \infty]$. Each Meridian node measures the distance d_j to a neighbour j and place the peer in the corresponding ring i such that $r_i < d_j < R_i$. Therefore, the neighbours are placed in concentric rings as function of the delay to the node. Each node keeps track only of k nodes per ring. The authors demonstrate in the paper that a choice of $k = O(logN)$ can resolve queries in $O(logN)$ lookups. In their evaluation for a Meridian overlay formed by 2000 nodes the authors use the following parameters: i^* (number of rings) = 9, k = 8, s = 2 and α = 1ms.

In case of having more than k candidate neighbours for a given ring, the node applies the following algorithm in order to select the most appropriate: it keeps track of the k primary ring members, but also a constant number of l secondary members learnt using gossiping. The ideal situation is having nodes in the ring covering as many as possible directions from the node. For this purpose, the node retrieves the measured distance between each pair of the k+l nodes in the ring. Based on this, it runs a local algorithm to select the k neighbours giving a better coverage of the ring. This is a costly process. It is also worth to note that although the authors claim the number of neighbours per ring is k, it is actually k+l.

In order to discover new nodes in the overlay network, Meridian relies on a gossiping protocol that works as follows. Each node A randomly picks a node B from each ring and send to B a gossiping packet containing one randomly selected node from each ring. Upon the packet reception, B probes the distance to A and all the nodes included in the packet. After sending the gossiping packet to all the selected nodes, A waits a predefined time to repeat the process. This process supposes a high overhead.

Once we have explained how Meridian system looks like, we next explain how it can be used in order to find a close-by server to a given client: a client C sends a closest node discovery query to a given Meridian node A. First, A determines its latency d to C. Then A asks all its neighbours with a latency from A between $(1 - \beta)d$ and $(1 + \beta)d$ to determine their distance to C. Finally, A forwards the query to the closest neighbour to C that repeats this process. The process continues until a Meridian node B is unable to find a closer node to C, thus being B the closest server to C. The described procedure has two main drawbacks:

- It incurs a high measurement overhead. For instance, in a Meridian overlay with 2000

nodes, the authors propose to use k = 8 servers per ring. Since the lookup complexity is O(logN) hops (i.e. around 10 hops in the example), then, we can conclude that around 80 servers would probe their distance to the client per each single query.

- The active probing of end-hosts has several problems: (i) most of the end-host in the current Internet are located behind NATs [CF07] and may not receive the probing messages; (ii) unexpected active probing of end-hosts may triggers intrusion detection systems and results in abuse complains [FLM06]. These problems dissapear if the probing is done from the client side.

Finally, it is worth noting that Meridian, rather than the Graph Representation Systems and the NCSs, is robust to the TIV problem and also to the variability of the properties of Internet paths since it relies on *live* measurements. Furthermore, it does not need any kind of dedicated infrastructure.

3.1.3.2 OASIS

OASIS (Overlay-based Anycast Service InfraStructure) is a shared infrastructure that offers an anycast service (i.e. location of a nearby replica server) to multiple services at the same time. Furthermore, the system is able to consider other parameters in the server selection such as liveness and load. It is worth noting that OASIS is publicly deployed on PlanetLab and has already been adopted by several services (e.g. ChunkCast, CoralCDN, Na Kika, ...). Furthermore, although we have classified OASIS within the On-demand Probing solution, it is a complex system that uses multiple techniques that goes further than a classical On-demand Probing system such as Meridian.

The system uses a two tier architecture: (i) There is a core of reliable hosts that implement the anycast service and (ii) the replica servers (from the different services registered in OASIS) form a loosely connected overlay network used to perform the network measurements. In order to avoid the heavy load produced by the classical On-demand Probing systems (e.g. Meridian), OASIS defines *stable network coordinates* for the replica servers. Each replica server is aware of its geographical coordinates (longitude and latitude). Furthermore, OASIS groups the end-hosts into network blocks that are identified by the geographical coordinate of the closest replica server in the overlay and the RTT measured from this replica server. In order to find the closest replica server to each network block, the replica servers in the OASIS system use Meridian. The authors claim that the measurement overhead is small since each network block is probed infrequently (e.g. once per week). On the other hand, the geographic coordinate scheme is much more robust and stable than the aforementioned virtual coordinates space. The information regarding the geographical coordinates of each netblock is stored in the OASIS core. Furthermore, each replica server stores in the OASIS core its geographical coordinates, its associated service and periodically updates its liveness and load.

In OASIS, when a client A launches a query to a OASIS core node in order find a close-by server: firstly, OASIS core assigns the IP address to a netblock and finds the geographical coordinate of the netblock. With this coordinate and the type of service, the OASIS core

finds the closest server[3] to the client. It is worth noting here, that in spite of the complexity of OASIS system the obtained replica server is the geographically closest server. It is well known that geographic distance is in many cases uncorrelated with topological distance in the current Internet. Hence, the selected server may lead to undesirable performance in some cases.

Following, we describe in more detail how the OASIS core works. The core is formed by a reduced number of nodes that are connected in a mesh. Each node in the core has an assigned id and the information stored in the core is distributed among the nodes using Consistent Hashing. This is, each object has an associated key and is stored in the nodes with the closest id to that key. For instance, a replica server of a specific service S registers in the core a tuple formed by a key k_S = hash(service name), geographical coordinates and load. This tuple is stored by the *Rendezvous Service* nodes, these are the nodes with the closest id to k_S. The same procedure is used to store the information regarding the netblocks. It is worth to notice that the Rendezvous nodes are a bootleneck of the system since all the queries for a given service are responded by them. The authors propose a locality technique to mitigate this problem. In OASIS the clients and the replica servers are redirected to the closest core node. Therefore, it is likely that clients and replica servers registered in the same core node are close to each other. Thus, when a core node receives a query from a client, it responds with its closest registered replica server (if there is not a registered replica server the core node uses the standard mechanism).

3.1.3.3 Discussion

The solutions presented in this section and the solution proposed in this Thesis address a similar problem. However, Meridian and OASIS present some aspects to be improved:

- The described systems have been designed for classical services where the replica servers are expected to be stable and permanently reachable. However, the usage of these solutions is not appropriate for applications with a high churn rate (e.g. peer-to-peer): On one side, Meridian would need a large measurement load to keep updated information in the overlay. On the other side, a high churn in the replica servers would produce an important overhead of the OASIS core since they must keep updated information regarding the available servers.

- OASIS utilizes a dedicated infrastructure of well connected core nodes that may affect the scalability of the system.

- Meridian incurs a high probing load per query. Furthermore, the probing is performed on end-hosts. This is a problem when the client is behind a NAT, in addition it can be intended as an intrusion.

[3] OASIS could use any selection algorithm using as input parameters: distance, load and liveness.

3.2 Related Work in Fairness in Chord and DHTs

Our proposed solution is based on a Chord DHT. Thus, the servers belonging to our system suffer from an extra overhead added by their participation in the Chord DHT. Since the size of the objects stored in our system is small (e.g. Server's AS number, country and continent) routing dominates the cost of publishing and retrieving them [GS05].

Therefore, we have proposed a solution that balances the routing load among the nodes in Chord DHTs. To the best of our knowledge there is any other previous proposal specifically addressing the load balancing of routing in Chord. Most of the previous work focuses on balancing the number of stored objects. In this section we present the most relevant bibliography regarding fairness in Chord and more generally in DHTs.

The root cause of unfairness in Chord comes from the ids assignment procedure. A node obtains a *node-id* by hashing its IP address (or any other attribute), thus the node is assigned to a random position in the ring. Furthermore, the node is responsible of the portion of the id-space between its predecessor id +1 and its own id. We name this area *the zone of responsibility* of the node. Therefore, all the objects whose object-key falls in the zone of responsibility of a given node are stored by the latter. In addition, all the finger search queries addressed to a key in the zone of responsibility of a node produce an inbound finger pointing to that node.

Hence, the id assignment procedure used in Chord (and also in other DHTs) leads to an unbalanced distribution of the size of the different zones of responsibility. Thus, if we assume that all the objects are equally popular and have the same size, those nodes having a larger zone of responsibility suffer from a higher load. In more detail, as shown in [SMLN$^+$03], some nodes suffer from an O(log N) imbalance (e.g. these nodes store O(log N) objects more than the average node).

The research community has proposed several solutions to mitigate this problem. They can be grouped into three families:

- *Virtual Servers (VS)*: In this case, the physical node is represented in the DHT by multiple instances (each one with a different node-id) named Virtual Servers. Thus, the zone of responsibility of a node is now the aggregate of the zones of responsibility of its VSs. The larger the number of VSs used the more equalized is the size of zone of responsibility of the nodes. The downside of the solution is that the overhead of a node increases with the number of VSs since the node needs to keep as many routing tables (i.e. successors and fingers) as number of VSs. Furthermore, the average number of hops per query increases since the overlay network becomes larger.

- *ID Reassignment*: In this case, the node is just represented by a single node-id in the DHT. In order to fairly distribute the id-space among the nodes, when a new node joins the network it does it by selecting the node-ID that minimizes the unbalance in the zone of responsibility's size distribution. Furthermore, the nodes periodically checks if they are using an appropriate node-ID, if not they pick up a new id that reduces the unbalance of the load in the network. The downside of these solutions

is the large amount of maintenance traffic generated because each node reassignment produces the exchange of stored objects among the affected nodes. This is specially harmful if the size of the stored objects is large. In other words, these solutions add an artificial churn to the system.

- *Other Solutions*: Here we include solutions that use a combination of techniques from both families [LS05]. We also discuss other solutions addressing the problem in a different way than the previous families. For instance, [SBKF07] present a load balancing solution for scenarios where the load in unfairly distributed among nodes (e.g. Zipf content popularity). The authors propose to remove the nodes responsible of popular objects from the routing tables, thus reducing their load in routing tasks.

Next we describe in more detail the most representative solutions within each family.

3.2.1 Virtual Servers

The usage of Virtual Servers was first introduced in [KLL$^+$97], the authors demonstrate that in scenarios where the objects are equally popular and the nodes use O(log N) virtual servers the imbalance factor can be reduced from O(logN) to O($\frac{logN}{loglogN}$) (This is from 20 to 4.63 if we consider a DHT with 1 million nodes). Furthermore, VSs can be used to deal with node heterogeneity. Those nodes having a higher capacity can be configured to use a larger number of VSs, by doing so they support a higher load than low capacity nodes (configured to use a smaller number of VSs). If a node becomes overloaded it simply reduces the number of VSs progressively until the load is adapted to its capacity.

The major problems of this solution are:

- **Problem 1**: The usage of V VSs multiply by V the number of connections to be established by the node, this is the number of fingers, predecessors and successors in case of Chord. Thus, the maintenance traffic is also increased by a factor V and the average path length by a factor of $O(logV)$. This problem is specially harmful in scenarios (e.g. Internet) where a large portion of the nodes are located behind Network Address Translators (NATs). In these scenarios, establishing a connection requires to implement and execute NAT Traversal protocols (e.g. STUN [RWHM03], TURN [RMM09], ICE [Ros07]) that makes the procedure costly. This becomes even worse if we consider the churn phenomenon. Let's consider the next simple example to put some numbers to the problem. In a basic Chord configuration each node establish connections to roughly O(log N) nodes[4] and receives connections from roughly the same number. If we consider an overlay with 1 million nodes, each node would have connections to roughly 40 nodes and the average path length is about 10 hops ($\frac{O(logN)}{2}$ [SMLN$^+$03]). Now, if we use the recommendation from [SMLN$^+$03] each node uses O(log N) VSs (i.e. 20 VSs in our example). Then, the network's size raises up to 40 million nodes and each node would need to keep connections to roughly 1000 peers.

[4]For this simple example we are only considering the fingers and not the successors.

This may be impractical in real implementations where NATs are involved. Finally, the average path length would increase up to 12 hops.

- **Problem 2**: The peers decide to increase or reduce the number of used VSs based on its local information. Moreover, in the case of being overloaded, they simply remove one (or more) VSs. Then, the objects left by the node pass to be under the responsibility of another node in an uncontrolled manner. This may lead to overload other nodes. Therefore, the method is not optimal in terms of load sharing.

More recent works aim to solve these problems. On one side [GLS+04, RLS+03, GS05, ZH04, ZH05] address the second problem. All these works propose solutions in which the nodes are aware of its own load but also the load supported by others. Thus, when one node is overloaded instead of simply releasing the load as in [SMLN+03, DKK+01] it transfers the load to an under-loaded node. Unfortunately, all these solutions suffer from the first described problem. On the other hand Y_0 [GS05] is to the best of our knowledge the only solution that address the first problem. Basically, the authors propose to assign to all the VSs of a given node close node-ids. Then the VSs can share the same finger table and the number of connection to be established is dramatically reduced. Unfortunately, this solution does not solve the second problem. Let's now describe the solutions in more detail.

In [RLS+03] *A. Rao et al.* classify each node as *heavy* or *light*. Let L_i denote the load of node i computed as the sum of the loads of all the VSs of node i. The authors assume that every node has a predefined target load T_i. A node is considered *heavy* if $L_i > T_i$ and *light* otherwise. The authors state that the binary model of the state of a node is simple and at the same time effective for the most common scenarios: homogeneous and heterogeneous nodes. In the case of homogeneous nodes, the aim is to divide the load of the system equally among all the nodes. Thus the target load is defined as the average load of the system (the authors propose to use random sampling to find out the average load of the system). In the case of a heterogeneous environment the objective is to define a target load proportional to the node capacity. Thus the target load of a given node i (T_i) with capacity C_i can be easily computed as $T_i = (\bar{L}/\bar{C})C_i$, where $\bar{L}$ and $\bar{C}$ are the average load and average capacity respectively (also random sampling can be used in this case).
The goal of this work is to define an algorithm that minimizes the number of heavy nodes by transferring load (i.e. VSs) from heavy nodes to light nodes. Given a heavy node h and a light node l, the authors define the *best virtual server*, v, to be transferred from h to l, that one satisfying:

1. Transferring v from h to l will not make l heavy.
2. v is the lightest VS that makes h light.
3. If there is not a virtual server whose transfer can make h light, transfer the heaviest virtual server v from h to l (fulfilling 1).

Furthermore, there are not VS transfers between either two heavy nodes or two lights nodes. The authors assume that the load of all virtual servers is bounded by a threshold equal to $\bar{T}$, if there is a VS overpassing this threshold the node must split this VS in as many

VSs as necessary to guarantee the threshold. Note that this splitting technique guarantees to avoid the presence of highly loaded VSs at the cost of adding extra VSs thus augmenting the Problem 1.
The authors present three schemes to make the VS transfer between nodes:

- *One-to-One Scheme:* Each light node periodically picks a random ID and perform a lookup operation to find the node that is responsible for that ID. If that node is a heavy node, then a transfer takes place between the two nodes if the previously described requirements are fulfilled. This scheme can be easily implemented in a distributed fashion. Furthermore, it is conservative since the heavy nodes do not perform probing task and if the system load is very high probing gets minimized. Finally, if the load of the nodes is correlated with the size of their zone of responsibility, since the probed ID is selected at random, it is likely that those most heavy nodes are rapidly found.

- *One-to-Many Scheme:* In this case, there exist *directories* in the DHT where the light nodes store information regarding their load. These *directories* are members of the DHT whose location is well-known. In this scenario, a heavy node h periodically samples the existing directories and sends its target load and the loads of all its virtual servers. The directory receiving this information decides the best virtual server to be transferred from h to a light node in this directory. The process is repeated until all the heavy nodes become light.

- *Many-to-Many Scheme:* This scheme is the natural extension of the two previous ones. In this case, both the heavy and the light nodes store their load information in the directories. Once the directory node d has information regarding enough nodes, it performs the reallocation algorithm and sends the solution back to the nodes. The solution specifies to each heavy nodes which VSs transfer to which light nodes. The authors perform a detailed simulation analysis of this scheme in [GLS$^+$04].

Y. Zhu et al. present a solution of VS transfer based on the previously introduced Many-to-Many Scheme [ZH04, ZH05]. In this solution a node i can be classified as : *heavy* ($L_i > T_i$), *light* ($(T_i - L_i) \geq L_{min}$) or *neutral* ($0 \leq (T_i - L_i) < L_{min}$). L_{min} is the load of the least loaded node in the system. Furthermore, the DHT nodes form a K-nary tree structure used to collect and propagate information regarding the nodes and system load. Based on this structure the authors propose a load balancing algorithm composed by the two following phases:

- Load Balancing Information (LBI) Aggregation: In this phase a node i reports to its parent node j in the tree $< L_i, C_i, L_{min,i} >$. These are the load of the node, the capacity of the node, and the load of the least loaded VS of the node respectively. Node j collects all the information from its children and compute $< L_j, C_j, L_{min,j} >$. This is the sum of the total load supported by j and its children, the sum of the total capacity of j and its children, and the load of the least loaded VS among those associated to j and its children respectively. As a result of this collection information procedure, the root node obtains $< L, C, L_{min} >$. This is the total load of the system, the total capacity of the system and the load of the least loaded VS of the system. The

root propagates this information to all the nodes using the tree structure. With this information each node is able to define its category (heavy, light or neutral).

- Virtual Server Assignment (VSA): The assignment is performed in a bottom-up fashion. A heavy node i selects a set of VSs that allow him to leave the heavy state, then i selects at random one of this VSs denoted as v and reports to its parent node in the tree $< L_{i,v}, v, ip_{addr}(i) >$. On the other side, a light node j reports its available capacity as $< \triangle L_j = T_j - L_j, ip_{addr}(j) >$. Every parent node receives the reports from all its children, if the total number of heavy and light children is over a threshold (e.g. 30) the node performs the VSs transfers, otherwise it forwards the information to its parent. The VSs transfers are performed in order from the heaviest VS to the lightest VS.

Moreover the authors argue that VS transfer can be very expensive if they are performed between nodes that are far away from each other, since the data must cross a long distance across the network. In order to avoid this, the authors apply locality techniques that allow physically close nodes to be also close in the id space of the DHT. The authors propose to use location aware techniques based on landmarks: each node measures its distance to N different landmarks and create a distance vector. Those nodes having similar distance vector are intended to be close to each other, thus they use close node-ids in the DHT.

This system adds an important degree of complexity (K-nary tree) and also an important measurement load (locality technique). Unfortunately the overhead added by this mechanism is not evaluated in the paper.

All the solution presented so far address the second problem presented above, however all of them suffer from the necessity of establishing a large number of connections that may be hard in real implementations. To the best of our knowledge the only VS-based solution addressing the first presented problem is [GS05]. *B. Godfrey et al.* proposes Y_0, an extended version of Chord also applicable to other DHT schemes. In Y_0 a node i having a normalized capacity c_i less than a given threshold γ_d is discarded to participate in the routing and storage tasks. It can only perform lookups. On the other hand a node j with a normalized capacity over the threshold uses $c_j\alpha$ VSs, where $\alpha = O(logN)$. The ids of the VSs are selected as follows: the node selects an initial point p_v as the hash(IP address), and then chooses one random ID within each of the $O(logN)$ consecutive intervals of size 1/N that are assigned to the VSs. Thus each node covers an area I_v of size $O(\frac{c_v logN}{N})$. Note that the areas covered by two different nodes can be overlapped. Moreover, when the number of nodes in the system (N) or the capacity of the node changes by a constant factor, the number of VSs is updated and new VSs IDs are calculated. However it must be noted that p_v remains the same. By selecting the VSs in a contiguous region of the id space the authors create a common finger table to be share by all the VSs. This finger table is created using p_v as reference id. This results in the node having a total number of $O(c_v logN)$ fingers. Therefore since the average capacity is 1, the average number of fingers for a node managing $O(logN)$ VSs is $O(logN)$, instead of $O(log^2 N)$ of the standard mechanism. Another important aspect to be considered is the number of successors s a node must keep. In the original proposal, a node having v virtual servers needs to keep connections to vs successors. Whereas, in Y_0 due to the

clustering of IDs, the VSs of a node share a large portion of their successors. Then the number of successors per node is $c_v s$. Finally, the routing is performed as in Chord: the nodes forward the queries to the closest finger (with lower id) to the destination key. Once the query reaches one of the VS inside of the area I_v covered by a given node, the destination can be either one of the node's VSs or one of the successors, thus the destination is reached in at most one hop.

Although this solution clearly reduces the overhead introduced by the use of VSs, it increases the probability of network partitioning. In Y_0 a given area of the Chord ring is now covered by a reduced number of nodes (even just one node in some cases), thus the churn can produce that all the nodes covering a part of the ring disconnect the system in a short period of time producing the network partition before the restoration mechanism can be performed. In order to avoid this problem the number of successors s must be incremented, thus adding an extra overhead and a higher number of connections to be maintained. Moreover, each time a node increases or reduces its number of VSs some zones of responsibility may be reassigned. This may lead to an exchange of stored objects among the involved nodes, thus increasing the overhead of the proposed solution.

3.2.2 ID Reassignment

The VS-based approaches suffer from an inherent overhead. Therefore other authors have proposed load balancing solutions in which the node uses a single id to participate in the DHT.

J. W. Byers et al. apply the *power of two choices* paradigm to DHT systems in order to fairly share the load of stored objects [BCM03]. The node-id is selected by using a defined function $h0$, however the object-key are computed by using d different hash functions (h_1,h_2,h_3,...,h_d). When a node desires to publish a new object x, it first computes the object-keys by hashing some property of the object (e.g. name) with the d hash functions. Then it looks for the d responsible nodes for those keys and retrieve their load information. Finally, x is stored in the least loaded node n. The other candidates store a pointer $x \Leftarrow n$. On the other hand, when a node wants to retrieve this same object x, it computes the object key by using one of the d defined hash functions. The search query reaches one of the responsible nodes for x. This node has x stored itself or it has a pointer to the node storing x. Then the look-up procedure adds in d-1/d percentage of the cases 1 extra hop. Furthermore, in order to make the system robust against churn the nodes must republish the objects periodically in the system and refresh the pointers. Moreover, the authors propose two mechanism for load balancing: *Load-stealing* and *Load-shedding*. In the former an underutilized peer p_1 seeks out load to take from heavy peers for which p_1 currently has a redirection pointer. Whereas, in the latter a heavy peer p_2 looks for light peers having pointers to him to transfer part of its load. Finally, the authors also propose a simple replication mechanism: the nodes instead of storing the object in just one of the d possible responsible nodes, do it in r of them ($d \geq r > 1$).
Note that this solution is designed for DHTs used to store relatively large objects. It does not solve the problem of routing unfairness in Chord.

D. R. Karger et al. propose a modification of Chord where each node has $O(logN)$ VSs but just one of them is active at any time [KR04]. The $O(logN)$ VSs have a fix id obtained by computing a hash operation over the node's IP address (hash(IP,1), hash(IP,2),..., hash(IP,log N)). Each node upon joining the DHT measures the size of the zone of responsibility associated to each one of its VSs, and activates the VS having the smallest zone of responsibility. The nodes repeat this procedure periodically and if there is a VS with an smaller zone of responsibility than the active one, they activate the former. Unfortunately, each time a node joins, reconnects or change its active VS, $O(loglogN)$ nodes need to modify their ids and also some objects needs to be transferred between the affected nodes. This generates a high overhead in networks suffering from a high churn rate. In order to mitigate this problem the paper proposes that the nodes cache the objects stored by its previously active VSs. Thus if the node needs to reactivate one of the previously active VSs, it already has stored some part of the objects associated to that VS. However, this technique incurs an important overhead if the stored objects are large. In summary, this solution seems to be appropriate for static environments, however in dynamic scenarios seems to create a large overhead.

Finally, K. Kenthapadi et al. propose an algorithm to select the id of a given node i joining the system [KM05]. i selects r ids from the id-space at random. For each id, i evaluates the length of the zone between the predecessor and successor of the id, the authors call this procedure *Random probing*. Furthermore, i learns the length of the v arcs in the vicinity of the selected random id. This procedure is named *Local probing*. The node selects the id that divides in two the largest measured zone. The authors claim that using $rv \geq cO(logN)$, where c is a small constant, the maximum difference between the largest and the smallest zone of responsibility is ≤ 8 times. As occurred with the previous proposal, this algorithm produces a high overhead in dynamic environments since each time a node connects to the system needs to probe at least $O(logN)$ nodes to select the appropriate node-id. Unfortunately any of the proposals has been evaluated under dynamic environments.

3.2.3 Other Solutions

In this section we present two types of works, those that mix VS and ID-reassignment algorithms and those that do not rely on any of these mechanisms.

J. Ledlie et al. present a complex algorithm that mix the usage of VSs with the ID-reassignment technique [LS05]. In more detail, each node when firstly joining the system receives k verifiable ids associated to a certified value x, this is k possible VSs. The ids are computed as hash(x+1), hash(x+2),...,hash(x+k). Furthermore, each node has a capacity c and a target load t. Upon the join, the node estimates the load associated to each one of its VSs. For its VS v it retrieves the load of the VS w (belonging to other node) responsible of the area of the id-space including v's id. The node estimates that the load to be supported by v would be the load supported by w multiply by the percentage of the id-space that v is going to take from w. The node repeats this procedure for the k VSs. After that, it selects the VSs to be activated in ascending order of estimated load, starting by the least loaded VS. It activates VSs until the target load t is achieved (if the target load is not achieved it

activates the k VSs). The authors propose an *active* version of the mechanism where the nodes periodically repeat the previous procedure and update the list of active VSs that offers the best load share to the system. This solution suffers from the aforementioned problems associated to the usage of VSs, furthermore its active version can incur high overhead since the id reassignment implies to transfer objects between nodes. This can be specially harmful in systems storing large size objects.

S. Serbu et al. present a load balancing solution for Pastry [SBKF07], however the scheme seems to be easily applicable to other DHT schemes, such as Chord. The basic idea is to release those overloaded nodes of their routing responsibility. For this purpose they are greedily removed from the routing tables of other peers. Every node keeps track of the load l_k for each node n_k in their routing table. In order to collect the load information each node, before forwarding or sending a message, adds itself and its load to the message, thus the i^{th} hop is aware of the load information of the previous i nodes. With this information the node checks if there is a node n_j within the routing table that can be substituted by another least loaded node n_k covering the same portion of the id-space. By applying this mechanism the heavily loaded nodes are removed from the routing tables. In order to avoid, new incomers being rapidly overloaded they set up their load as the average load of the nodes included in its routing table. The authors state that churn does not affect to load balancing, thus they do not evaluate they proposal under churn. However, it seems that the departure of a heavy loaded node may overload a node highly involved in the routing task, thus it would be interesting to evaluate the convergence of the proposed solution under churn conditions.

Part II

Dynamic and Location Aware Service Discovery Using DHTs

Chapter 4

Introduction

The Internet architecture has rapidly evolved in the last decade until become a complex system where is even difficult to depict the Internet map. Moreover, Internet is anymore a network dedicated to provide connectivity among hosts but the 1^{st} Service Platform of the world. If we focus on the services offered over Internet they have also evolved in the last decade. Now, most of them are global services accessed by million of users spread across the world (e.g. Web portals, Video Streaming Portals). Then, the concept of server is anymore a single machine offering a specific service (e.g. Web Server) but a large amount of machines organized in centralized or distributed architectures. For instance: server farms, distributed data centers, Content Distribution Network (CDN), etc. Furthermore a new paradigm of services, the peer-to-peer services/applications, have emerged and become the most popular applications in the Internet [CAC07]. In this new paradigm the end-hosts (e.g. home users) are at the same time clients and servers. Then, the servers (end-hosts acting as servers) of P2P applications are not *always-on* hosts as expected from a server. Finally, the behaviour of the users or service's clients has also been modified. Now, a single user accesses the services from many different geographical and topological locations across the time. Furthermore, the rapid evolution of wireless technologies (e.g. WiFi or Cellular Networks) has opened the possibility to a new model of connectivity in which the users have permanent access to Internet while moving.

In this Thesis we focus on a small corner of the presented big picture. In the current (and also in the future) Internet there exist some worldwide deployed services where it is crucial for the client to connect to a server that guarantees the required QoS of the communications. This means a bounded delay and enough available resources. In fact, this can be achieved by choosing a topological close-by server to the client with enough resources (e.g. available bandwidth). Examples of services in need of these solutions are: (i) *Content Distribution Networks (CDNs)*. They are used for Video Streaming applications (e.g. YouTube), web pages, etc. The CDN replicates the content (e.g. Video) in multiple servers across the world, thus the clients must be redirected to a close-by one. This improves the performance of the content transfer and reduces the cost of the content provider. (ii) *Online Gamming Applications*. In this case, the user's experience highly depends on the delay to the server.

Typically, the actions (e.g. shooting) are sent to the server that compiles the different users' actions and send back the result. Thus, it is desirable that the users are close to its server in order to avoid de-synchronization problems that affect to the user's experience. (iii) *P2P file sharing applications*. In this case, the user can typically selects to download a file from different sources. Then, it is desirable to select a close-by source. (iv) *NAT Traversal Servers*. The P2P applications, such as file-sharing (e.g BitTorrent) or VoIP (Skype) present a large portion of their users behind a NAT. In some cases the users cannot traverse the NAT and need to use a third entity named Relay (in the case of Skype are also named Super Nodes) to establish and receive connections. It is desirable to use a close-by Relay to one of the users involved in the communication, thus reducing the end-to-end communication delay.

The high interest of the problem has attracted the attention of the research community and some solutions to discover a close-by server to a client have been proposed. Furthermore, some other proposals that do not address specifically this problem could also be adapted to solve it[1]. However, each one of the previous proposals has some disadvantages, shown in Table 6.1, that we would like to eliminate (or at least mitigate).

Therefore, the first objective of this Thesis is the design of a solution for dynamic and location aware server selection that fulfils the following requirements:

1. Fully Distributed
2. Do not use end-host probing
3. Robust to Triangular Inequality Violation
4. Robust to Churn
5. Robust to User Mobility
6. Low Measurement Overhead
7. Low Maintenance Overhead
8. Simplicity
9. Server selection based on distance/delay and other parameters such as server capacity and availability
10. Offer an Anycast service (i.e. to be able to support close-by server discovery for multiple services at the same time)
11. ISP friendliness

To achive the defined objective we propose the usage of a Distributed Hash Table (DHT) formed by all (or a subset of all) the servers offering a specific service (e.g. NAT Traversal

[1]For a detailed description of all these solutions we refer the reader to Section 3.1.

Service, Online Game Service, SIP Proxies, etc...) where they register their location information. Thus, the clients are able to query this DHT in order to find a close-by server (i.e. one with a close location to the current client location) that allows preserving the QoS of the communications. In more detail the solution works as follows: The servers of a specific service S form a DHT where they store their topologically (e.g. AS number) and/or geographical information (e.g. Country). Furthermore, each client using service S is connected to at least one of the servers in the DHT. If a given client C is using a server S_1 (connected to the DHT) that is offering a bad performance to C's communications, then C starts the procedure to find a new server. For this purpose, it sends a *find server* query to S_1 that infers the location information of C based on its IP address (e.g. C's AS number and C's Country). With this information, S_1 composes one (or several) query(ies) and sends it into the DHT. As result, S_1 obtains a list with the IP address of those close-by servers to C. The list is forwarded to C that selects a server based on a predefined criterion, for instance, the server offering the lowest delay. For this purpose C measures the delay to each one of the servers in the list and connects to the one offering the lowest delay. Other criteria could be the server with most available bandwidth, the server with most available resources (CPU, memory, etc), random selection, etc..

We demonstrate the simplicity, practicability and functionality of the proposed solution by applying it to two different services: NAT Traversal Server/Relay Discovery [CCCA$^+$10] and Home Agent Discovery in Mobile IP scenarios [CCA$^+$09, CACDP$^+$09, CGC$^+$09, CCUnG07, CCG$^+$07]. The former validates the solution in a dynamic scenario where the servers are typically home users running a P2P application. Whereas, the latter validates the proposal in a classical client/server scheme where the servers are hosts with a dedicated function and are likely to be *always-on*. Next we briefly discuss the importance of Location Aware Server Discovery in each one of these services:

- *NAT Traversal Service:* the Network Address Translator (NAT) has been adopted as the de-facto standard by ISPs and Companies to mitigate the problem of the rapid IPv4 addresses depletion. The usage of NATs allows multiplexing a single public IP address in order to gain Internet connection from multiple hosts. The NAT and NAT-PT [AD07] were designed at the moment when the client/server paradigm was the unique service model in the Internet. The huge majority of end-hosts (e.g. residential hosts) were located behind NATs and were typically clients, thus being responsible of opening the connections to the servers. Whereas, the servers were a reduced number of machines with public IP addresses. In those years, only few protocols had problems with the presence of NATs (e.g. protocols including the IP address information in the application layer payload). However, in the last decade a new paradigm of services, the P2P services/applications, showed up and rapidly gain importance. In these applications, a end-host plays the role of server and client at the same time, thus opening connections but also receiving incoming connections. Unfortunately, the NATs do not allow incoming connections. Therefore, the presence of NATs was affecting some of the most used applications such as P2P-VoIP Applications (SIP-based applications or Skype) and P2P File-Sharing applications (BitTorrent or Emule). The research community rapidly reacted and designed NAT Traversal protocols (e.g. STUN [RWHM03] or TURN [RMM09]). These protocols allow the client behind a NAT to discover

which type of NAT it is behind of. For this purpose the clients use a NAT Traversal Server. The worst case happens when two nodes behind NATs wish to establish a connection, then they discover the NAT they have in front and if possible they connect directly. Unfortunately there are some types of NATs (e.g. symmetric NATs) that avoid that the communication can be directly established. In this case, the nodes use a third entity named Relay (that typically is the same NAT Traversal Server) to establish a relayed communication. It is worth noting that this situation can be also produced by the use of firewalls, quite extended in the companies to control both the outgoing and incoming traffic.
Then it is desirable that each client connects to a topologically close-by Relay with enough available bandwidth. This guarantees the QoS of the communications. Our solution achieves this aim in a simple and practical way.

On the other hand, the ISPs have started to be worried due to the large amount of transit traffic generated by the P2P applications that is out of their control. This traffic is increasing the costs of the ISPs, thus they have started to throttle the traffic of some P2P file-sharing applications such as BitTorrent [Coh03]. This directly affects the performance experienced by the users of these applications. In response, the developers of P2P applications have designed some techniques to avoid the filtering of ISPs. In order to bring peace to this conflict, the research community has proposed solutions whose purpose is to keep as much P2P traffic as possible inside the ISP's network. This reduces the traffic going to the transit links and does not affect the user's experienced performance. These techniques are known as *Locality or ISP friendly* solutions. Since our proposal is likely to be used by P2P applications, it is desirable to design it as an ISP friendly solution that minimizes the transit traffic produced by the Relays of a given ISP. Our solution also addresses this issue.

- *IP Mobility Service:* Within IP Mobility we find Terminal Mobility and Network Mobility. In both cases, the mobile terminal or the mobile network connects to other nodes in the Internet (named Correspondent Nodes) through a third entity called Home Agent. The Home Agent is typically located in the Administrative Domain of the mobile terminal/network, this is its ISP or other sub-domain inside the ISP. It could happen that the mobile terminal/network and the correspondent node are located within the same Autonomous System (AS) from an European country, whereas the Home Agent is located in the US. Then, the communication crosses a transoceanic link twice, thus suffering from an unnecessary high delay that may not be supported by delay sensitive applications such us multimedia applications (e.g. Voice or Video). This *Triangular Communications Problem* is well-known and has been addressed by the research community. A first family of solutions proposes to use the Home Agent only in the communication establishment phase; afterwards the packets flow directly between the Mobile Node and the Correspondent Node. This solution is viable in Mobile IPv6 (MIPv6), unfortunately it does not work for either IPv4 or Network Mobility (NEMO). A second family of solutions proposes something similar to Cellular Networks' Roaming. In this case, there are several Home Agents distributed across the world, thus the client could be connected always to a topologically close Home Agent that guarantees a low delay in the communications. In this scenario, it is necessary to

define a scalable and dynamic solution in order to discover the topologically close-by Home Agents. This is exactly what our proposed solution does.

Chapter 5

Description of the proposed solution

In this section we describe the proposed solution using as example an abstract service that we refer to as Service S. As stated before the main objective of our solution is finding a close-by server that guarantees the QoS of the communications. This is, a bounded delay lower than a given threshold (e.g. 150 ms in VoIP) and enough available resources (e.g. bandwidth). Next we enumerate the different logical components of the proposed solution:

1. *Overlay Topology*: How looks like the topology created by the servers and the clients.
2. *Join/Departure Procedure*: How the clients and the servers join and leave the system.
3. *Location Information Storage*: Which type of location information can be stored in the system and how it is stored.
4. *Server Discovery Procedure*: How the client discovers a close-by server.
5. *Server Selection Procedure*: How the client selects a server to join among those that are close.
6. *Fault Tolerance Mechanisms*: The needed mechanisms to make the system robust.

In the rest of the section we describe in detail each one of the aspects of our solution introduced above.

5.1 Overlay Topology

Our architecture is composed by a Distributed Hash Table (DHT) formed by all (or a subset of all) the Servers offering a given service S. We consider Chord as the default DHT to design our solution, although any other DHT scheme would work similarly. Chord is one of the most representative DHT schemes and has been implemented as part of different real

systems. In addition it has been recently selected as the mandatory DHT scheme to be used by the P2PSIP WG.

In our solution, each Server has a Peer-ID that determines its position in the Chord ring. The Peer-ID is obtained by computing the hash of the Server's public IP address. Furthermore, the servers store their location information in the DHT. On the other hand, the clients of service S are connected to (at least) one server belonging to the aforementioned DHT. Figure 5.1 depicts the described topology.

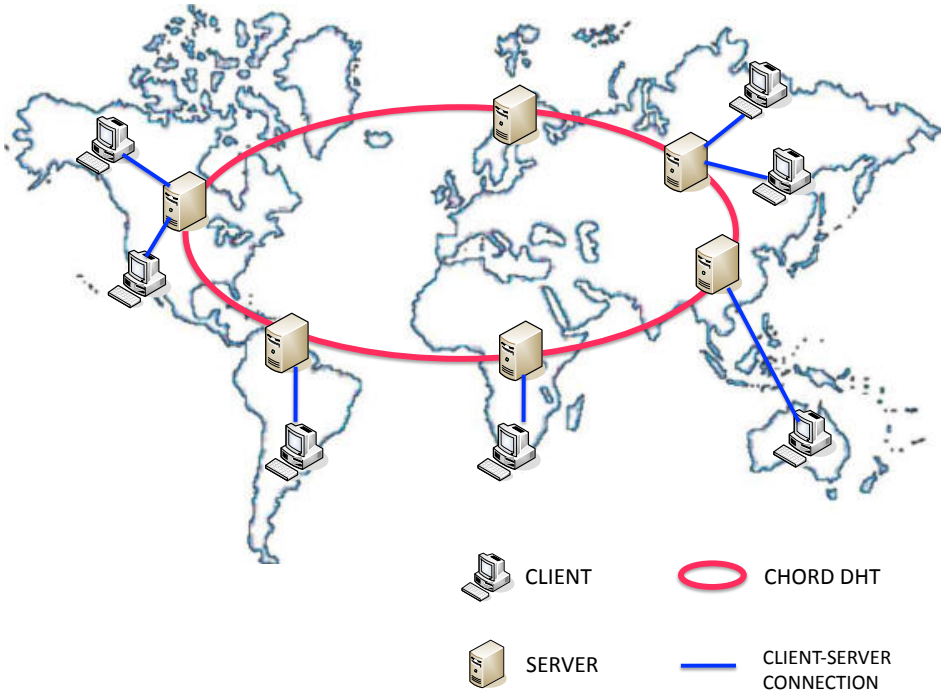


Figure 5.1: *Overlay Topology of the proposed solution*

5.2 Join/Departure Procedure

When a node is activated, it firstly identify if it is either a *server* or a *client*. In some services, the node is by definition a client or a server (e.g. Mobile IP). However, in other cases the node could play the role of either a client or a server depending on the circumstances (e.g. NAT Traversal Servers in applications such as Skype). In this case, we propose to use a simple rule: if the node is reachable (i.e. it is not behind a NAT) it acts as a server, otherwise, it is a client.

If a given server S_1 desires to join the system, it has firstly to join the DHT. For this purpose, if it is the first join, S_1 must contact an initial bootstrapping node (this can be for

instance a node from a list of well-known bootstrapping nodes). After the first join, S_1 learns the IP address of other servers, then in the subsequent join procedures, S_1 uses one of these learnt IP addresses to join the DHT (in the case of any of them being available, S_1 uses again the well-known bootstrapping nodes). S_1 provides its Peer-ID to the bootstrapping node, that answers with the IP address of the successor (s) of S_1 in the Chord ring. Then, S_1 contacts s in order to obtain its predecessor, its successors and its initial fingers. Afterwards, S_1 receives from its successor and its predecessor the keys for which it is now responsible. Finally, S_1 stores its location information in the DHT as described in 5.3. This concludes the join procedure. When a server S_1 decides to leave the system, it initiates the departure procedure. Firstly, it leaves the DHT. For this purpose, S_1 informs its neighbours and assigns to the appropriate neighbours the responsibility of its keys. S_1 also removes its stored location information from the DHT. In addition, if a given client C is connected to S_1, the latter informs the former about its departure, thus C can selects a new server. S_1 does not leave the system until all its clients switch to a new server.

On the other hand, if a client C wants to join the system, it has to join one of the servers belonging to the DHT. For this purpose, if it is the first join, C contacts one of the initial bootstrapping nodes and ask for a server. The bootstrapping node assigns to C a random server from the DHT. C learns the IP address of several servers across the different sessions, thus in subsequent join procedures C tries to connect to any of the learnt servers. If none is available it uses again the bootstrapping nodes. In the departure process, C informs the server it is connected to that releases the resources assigned to C.

5.3 Location Information Storage

The final aim of our system is to discover a close-by server to the client that guarantees a good QoS. For this purpose, the servers joining the DHT register their location information in the DHT. This info can be of two different types:

- Topological Information: Autonomous System (AS) number, IP Prefix.
- Geographical Information: Geopolitical (City, Region, State, Country, Continent), GPS coordinates, Geographical Coordinates (Latitude and Longitude).

Depending on the type of service, the information registered in the DHT is a different combination of the previous ones.

The location information is registered in the DHT as tuples $< key, value >$. For instance, a given server S_1 in order to store its AS number (AS_{S_1}) computes an associated key (K_{AS,S_1}) as the $hash(AS\ number)$. Then, S_1 stores the tuple $< K_{AS,S_1}, AS_{S_1}, S_1's\ IP >$. This tuple, is stored by the successor of K_{AS,S_1} (i.e. the node with the immediate higher Peer-ID to this key). This procedure can also be applied to IP prefixes of different lengths (key = hash(IP Prefix)), and the Geopolitical information[1] (key = hash(City Code), key = hash(Region Code), key = hash(Country Code), etc).

[1]The different codes are obtained from ISO 3166.

On the other hand, the GPS and the Geographical coordinates are special cases. A query including coordinates is typically a *range query*. This is, it looks for the servers located in a given area defined by a set of coordinates (included in the query). Therefore, in this case the key is formed directly by the coordinates (without applying on them any hash operation). Thus, the contiguous coordinates are stored in the same or in consecutive nodes in the DHT. It is worth noting that some DHT schemes such as CAN [RFH+01] (where the zone of responsibility of a node is a n-dimensional Cartesian area) are more appropriate in this specific case.

5.4 Server Discovery Procedure

In the described architecture, a given client C is connected to a server S_1. In case S_1 does not fulfil the QoS requirements (e.g. bounded delay and enough available resources), the client launches the *Server Discovery Procedure*: C sends a *find server* query to S_1, then, S_1 infers the location information of C based on C's IP address. For instance, C's AS number, country, region, city, etc. With this information, S_1 obtains the search keys by computing the necessary hash operations. Afterwards, S_1 launches one or multiple queries in order to find servers located close to C (e.g. servers in the same AS and/or same city and/or same country). For this purpose two type of procedures can be applied:

- *Sequential Search:* S_1 launches sequentially the search queries. If one of the queries succeeds then it stops the search procedure. Let's assume that S_1 has computed the AS number, city and country keys associated to C's IP address. Then, it first sends a query within the DHT asking for the AS number key. Therefore, if there is at least one server located in the same AS as C, the query succeeds and S_1 obtains as answer a list with the IP address of all the servers located in C's AS and the search procedure ends. However, if the query does not succeeds (i.e. there is not a server in C's AS), S_1 repeats the process with a second key (e.g. city key), if again the search fails it repeats the process with the third key (e.g. country key) and so on. If none of the queries succeeds C keeps connected to S_1 and repeats the server discovery procedure after a preconfigured time. In Chapter 7 we apply an adapted version of the Sequential search to the problem of NAT Traversal Servers discovery named *Gradual Proximity Algorithm*.

- *Combined Search*: S_1 launches in parallel the different search queries. It locally computes the responses in order to find the server that fulfils a larger number of search criteria. If we consider the previous example, S_1 sends in parallel the three search queries (AS number key, city key and country key). As result, S_1 obtains three list of IP addresses with the servers located in the same AS, the same city and the same country as C respectively. Then, S_1 selects the best server(s) based in a predefined selection criterion, for instance: servers in the same AS and the same city and the same country as $C >$ servers in the same AS and the same city as $C >$ servers in the same AS and the same country as $C >$ servers in the same city and the same country as $C >$ servers in the same AS as $C >$ servers in the same city as $C >$ servers in the

same country as C. Note that $>$ indicates higher preference.
As in the previous case if all the queries fail, C keeps connected to S_1 and repeats the server discovery procedure after a preconfigured time.
The combined search gives more accurate results at the cost of a higher overhead.

Finally, S_1 sends the list of selected servers to C. This ends the server discovery procedure.

We have stated before that a server must be able to infer the location information of a given client from its IP address. There are several commercial and open source solutions that can be used for this purpose [who, Max, geo, hos, ip2, rou].

5.5 Server Selection Procedure

As result of the Server Discovery Procedure a client obtains a list of the IP addresses of close-by servers. Among these servers, the client has to select one. This selection can be done based on several criteria. Next, we enumerate some examples:

1. Economy criterion: the client sorts the servers from the cheapest to the most expensive and selects the cheapest. This criterion can be used in those services that have an associated cost.
2. Security criterion: the client ranks the servers based on the offered security and selects the one providing the highest security in the communications, e.g. a certificate from a trusted company.
3. Performance criterion: the client ranks the server based on a performance metric and selects the one offering the highest performance. Since, the performance can be measured using different metrics, we next show some examples.
   - Delay: the client selects the server offering the lowest delay. For this purpose, the client measures the delay (e.g. RTT) to each one of the servers, and selects the one with lowest delay. This criterion can be used in those delay sensitive services such as VoIP.
   - Bandwidth: the client selects the server with more available bandwidth. For this purpose, the client queries the different services to find out their available bandwidth. This criterion can be used in those high bandwidth consumption applications such as file sharing.
   - Computational Capacity: the client selects the server offering the highest computational capacity. For this purpose, the client also queries the different servers to discover their capacity. This criterion can be used in grid or cloud computing systems.

   It must be noted that a combination of these criteria can be used. For instance, in the proposal described in Chapter 7 we propose to select the closest server having enough

available bandwidth. Whereas, in Chapter 8 the client selects the closest server having a valid certificate.

By combining the Server Discovery and Server Selection procedures, the client achieves the defined aim: *to dynamically discover and select a close-by server that guarantees the QoS of the communications*.

Figure 5.2 presents an sketch of the Server Discovery and Selection Procedures.

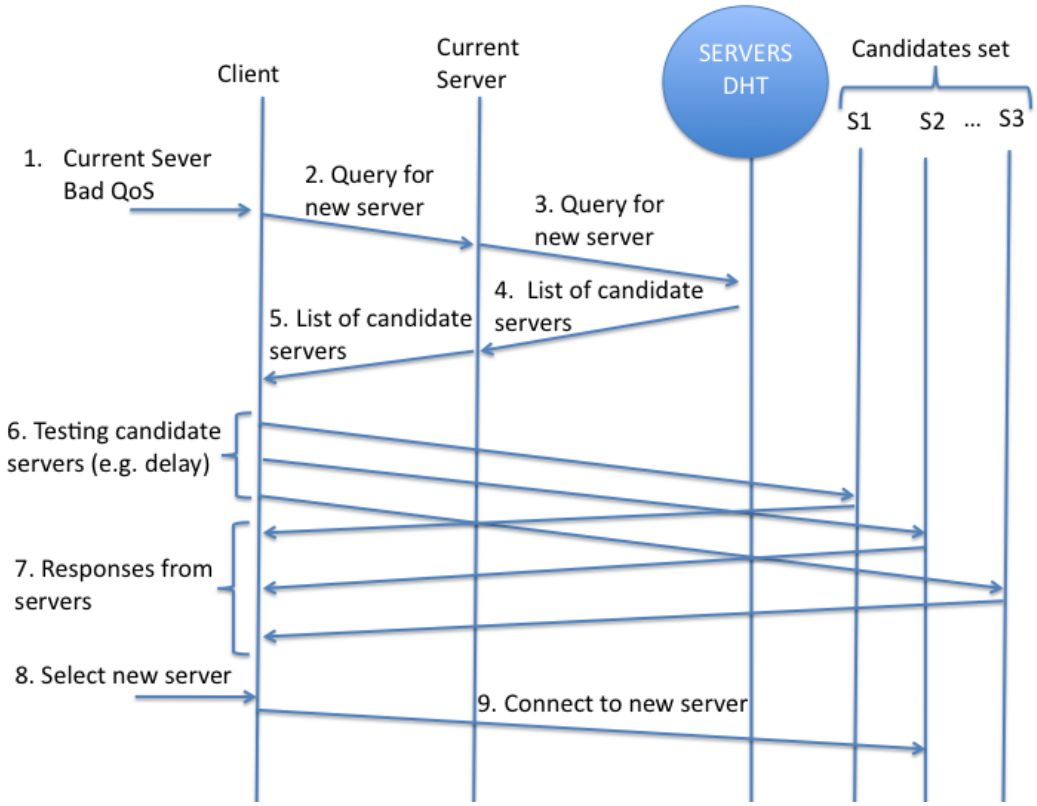


Figure 5.2: *Proposed solution basic functionality: Server Discovery and Selection Procedures.*

5.6 Fault Tolerance Mechanisms

The proposed architecture is dynamic since the servers and the clients can connect and disconnect at any moment. We have explained above the graceful join and leave procedure of both clients and servers. However, there could be some cases in which the nodes can disruptively leave the system. In these cases we need to define the necessary mechanisms to avoid: (i) inconsistency and lose of information in the DHT, (ii) resource infra-utilization in the servers and (iii) communication interruption.

When a given server S_1 disruptively leaves the DHT, the location information of S_1

stored in the DHT is anymore useful and could produce mistakes in the server selection (e.g. a client may select S_1 as server when it is actually off-line). Furthermore, the keys stored in S_1 may be lost when the server suddenly disconnects. To mitigate the former problem, the location information registered by a given server has a time-out associated. If the time-out expires without an update, the responsible node(s) of the information removes it. This bounds the time a server is announced as available by the DHT while being off-line. Furthermore, our system relies in *live* probing so the client can easily identify that the server is not available and connect to a different one. On the other hand, to eliminate the information loses we use replication techniques. This is, a given key is stored by r successors rather than just one successor. Thus if one of the key's successors unexpectedly leaves the ring the key is still accessible in the other r-1 successors.

Furthermore, one or more clients may be using a server that disruptively leaves the system. If this happens, the client needs to reconnect to a new server as soon as possible. For this purpose, the client always store the last list of ranked servers, then if the server it is connected suddenly leaves the system, the client reacts by selecting the highest ranked node (that is active) from the list. If there is any active server from the list offering the required QoS, the client triggers the Server Discovery Procedure. An alternative mechanism is that each client is concurrently connected to two (or more) servers, thus in case the primary server fails the client can immediately switch to the secondary one.

On the other hand, if a client using a server leaves the system disruptively, the server should release the resources assigned to that client in order to be used by another one. For this purpose, the server configures a time-out for each client. If a given client's time-out expires without having news from this client the server release the resources assigned to it.

The described mechanisms can be partially modified and adapted for each specific case.

Chapter 6

Discussion and Comparison with the Related Work

In this section we demonstrate that our system fulfils the 11 design requirements we have defined. Furthermore, we compare it with the previous proposed solutions demonstrating that our solution is the most complete one according to these requirements. We refer the reader to Section 3.1 for a detailed description of the different solutions discussed in this section. A summary of the discussion can be found in Table 6.1.

Fully Distributed Our architecture is fully distributed and does not rely in either Vantage Points to perform measurement tasks as in the Graph Representation Solutions or a Core Infrastructure as in OASIS. This eliminates any constrain to the system scalability. Other systems such as Meridian [WSS05] or the distributed NCSs [DCKM04, LGS07, AL09] also present this property.

Do not use end-host probing In our solution, the measurements are done from the client side to the server side. Other solutions such as Meridian or OASIS do it in the other way around. This has two problems that our solution eliminates: (i) the end-host probing may not work in those clients behind a NAT, (ii) the end-host probing may activate the intrusion detection alarms. Other solutions such as the NCSs [DCKM04, LGS07, AL09] also have this property.

Robust to the TIV Problem and Network variability Our solution relies on actual and *live* measurements instead of estimation. This solves two problems: (i) since we rely on actual measurements rather than estimation techniques we do not suffer from the TIV problem and (ii) since we use *live* measurement, we are aware of the network performance at the moment of selecting a server. Thus, we are robust to network variability (e.g. Congestion, route modifications, etc.). Unfortunately, all those solution using estimation techniques (Graph

Representation, NCSs and OASIS) suffer from these problems. On the other hand, Meridian is the unique proposed solution using actual and *live* measurements, thus the unique presenting this property.

Robust to Churn Our system is able to react to servers and clients churn using the Fault Tolerance Mechanism previously described. To the best of our knowledge none of the previous works discusses either how to react to churn or what the effects of churn are. We believe that an effect in all the systems is the increment of the measurement overhead. Let's focus on the solutions specifically addressing the problem of close-by server selection: Meridian and OASIS. In both cases it seems that the client churn is not a big issue, however the server churn produces a heavy overhead. In Meridian, a high Server Churn produces a high maintenance overhead, since the overlay structure is complex and even without churn suffers from a high load. On the other hand, in OASIS each time a Server leaves or join the network, the Core must update the information regarding Servers availability, thus incurring a high maintenance overhead. Finally, it is not clear how a client using Meridian or OASIS reacts when the server it is connected to leaves the system.

Robust to User Mobility In the current Internet it is likely that a user moves and changes its point of attachment to the Internet (i.e. its IP address). In our system if this happens, the client realizes that the server it is using is anymore valid and launches the server discovery procedure. This is again not discussed in any previous work. However, we believe that other systems such as Meridian or OASIS can react to the user mobility in a similar way. Also the Graph Coordinates Systems can easily react to this problem since their measurements techniques consider IP prefixes to infer the delay between two points instead of individual users. On the other hand, the distributed NCSs use individual measurement per node, thus when a node disconnects and connects again in a different position, it must start the process to find out its location in the virtual graph. Some solutions, such as Htrae or Pyxida propose that the node bootstraps on each session with the coordinates from the previous session. This is problematic in the described scenario. Furthermore, it is not clear if these systems are adaptive to mobility scenarios where the user changes its location while being connected to the system.

Low Measurement Load In our system the measurement process occurs only in the Server Selection Procedure. The client queries the *candidate* servers (obtained from the Server Discovery Procedure) in order to select one of them. For instance, if the client is interested in connecting to the closest server it measures the RTT to each one of the candidates and selects that one offering the lowest RTT. Therefore, the measurement load is reduced to the number of servers to be queried (S_q) during each server selection procedure.

One of the advantages of this mechanism is that the measurement load is adapted to the system necessities. This is, in static and stable environments where the servers are *always-on* and the clients do not change their location, a client C selects a server S_1 that guarantees the QoS in its first session. In the subsequent sessions it re-connects to S_1 and only in the case that S_1 is overloaded or unavailable, C looks for a new server. On the other hand, in an

extremely dynamic scenario with mobile users and high churn of both servers and clients, our system is still able to offer accurate results at the cost of $\sum_i S_{q_i}$ where $i \in [1, 2, ..., N]$ (being N the total number of searched servers). Furthermore, we can define a parameter $S_{q_{max}}$ to indicate the maximum number of servers to be queried by each client (e.g. 20). By doing so, if a client receives a list of servers with more than 20 entries, it selects 20 at random and queries them. If there is at least one server offering the expected QoS the client connects to it. Otherwise, the client explores the rest of servers from the list in sets of 20 until obtaining one guaranteeing the desired QoS.

Therefore, our solution is inherently adaptive to the system demand. This is also a characteristic of Meridian. On the other hand the Graph Representation Systems, the distributed NCSs and OASIS use a static and periodic measurement mechanism. For instance, in the case of the NCSs the nodes are constantly measuring their positions independently of the dynamism of the system. In stable systems, this may produce and unnecessary extra overhead whereas in highly dynamic scenarios the nodes are not able to update its position rapidly enough. It is worth noting here that Pyxida uses a very low measurement overhead since it piggybacks the measurement task into other messages. However, as explained in Section 3.1 this affects the accuracy of the measurements.

Let's now compare our solution with Meridian. For this purpose we use the same example given in the Meridian paper, an overlay with N = 2000 servers and k = 8 servers per ring. The lookup complexity in Meridian is $O(logN)$ (around 10 hops in this example). In addition, on each look-up hop k servers must measure their delay with the client. Therefore, Meridian needs to perform 80 probes to find a close-by server in the proposed example. On the other hand, in our solution we need to perform at most S_q probes. Thus, for any $S_q \leq 80$ our system suffer from a lower load measurement than Meridian. Furthermore, Meridian uses end-host probing whereas in our solution the probing is performed from clients to servers.

Low Maintenance Overhead In our solution, the major portion of the maintenance load is that one associated to the DHT. The DHTs have been demonstrated to be a very scalable solution requiring a low maintenance overhead. Clear examples are KAD and Azureus, a DHT used for file-sharing on the Emule overlay and a DHT used for tracker-less functionality of BitTorrent respectively. Both of them count with million of concurrent active users [SENB07], [ZDWR]. Furthermore, most of these users are typically home users with relatively low resources (e.g. bandwidth).

The proposed solutions using a infrastructure such as the Graph Representation Solutions or OASIS requires human intervention to develop some parts of the maintenance task. This incurs a high cost. On the other hand Meridian proposes a complex overlay system that requires the nodes to be continuously testing other nodes in order to select the best neighbours. Finally, the distributed NCSs require a low maintenance overhead.

Simplicity Our solution is simpler than any of the previous proposals. The Graph Coordinates Systems and OASIS uses a measurement infrastructure and a core infrastructure

respectively. This makes the system more complex since these infrastructure nodes must be configured and maintained. The distributed NCSs are simple in essence, however in order to address some of their described problems they need to implement sophisticated mechanisms that make them more complex. A clear example of this is Htrae [AL09]. Finally, Meridian relies in an overlay of servers, this is conceptually similar to our solution. However, the Meridian's overlay structure is much more complex than a simple DHT that is what we propose in our solution.

Server selection based on distance/delay and other parameters such as server capacity and availability To the best of our knowledge OASIS is the sole solution that allows to select a server based on other parameters in addition to delay. Our solution also offers this possibility, since a client when querying the candidate servers asks for the needed parameters, e.g. available bandwidth or available resources (memory, cpu, etc). Thus the result in both cases is a close-by server with available resources.

Offer an Anycast service (i.e. to be able to support close-by server discovery for multiple services at the same time) To the best of our knowledge OASIS is the unique solution offering an Anycast service. This is, the infrastructure is shared by servers belonging to different services, thus clients from different services can find a close-by server using OASIS.

We can easily extend our solution to provide an Anycast service. For this purpose, when a server registers its information in the DHT, it stores also the *type of service*, S. For instance, a given server S_1 registers the information regarding its AS number in the following way: S_1 computes a key with information regarding the AS number and the type of services as $K_{S,AS,S_1} = hash(AS\ num|S)$. Then, the tuple to be stored is: $< K_{S,AS,S_1}, S, AS_{S_1}, S_1'sIP >$. The only requirement is to agree a common nomenclature for the *type of service*. For instance, the name of the service can be used: e.g. *dns* for DNS servers, *nat traversal* for NAT traversal servers, etc.

ISP Friendliness Most of the close-by server discovery solutions are likely to be applied in P2P applications. Examples of this are Pyxida and Htrae. As stated before in this Thesis, the ISPs are worried due to the high traffic load generated by certain P2P applications (e.g. BitTorrent) that escapes to their control. A large amount of the P2P traffic goes to the ISPs' transit links increasing their operational costs. Some ISPs have started to shape and throttle the BitTorrent and other P2P applications' traffic. In response the P2P application programmers have implemented countermeasures to these techniques. The research community is trying to bring peace to this conflict. For this purpose, the researchers have proposed some solutions that aim to keep as much P2P traffic as possible inside the client's ISP. These solutions are know as *locality solutions* and its main rationale is to bias the neighbour selection procedure of the clients in order to make preferable the nodes within the same ISP as the client. This concept has not been previously discussed in the arena of close-by server selection. For instance Pyxida is used as a component of Azureus, one of the most used BitTorrent clients, but do not consider this aspect.

We have decided to include the concept of ISP Friendliness in our solution. For this purpose we prefer always those servers located in the same AS as the client. Then, the first search query in the *sequential search* looks for servers within the same AS as the client. Whereas, in the *combined search* the selection criterion always prioritizes the servers within the same AS as the client. It must be noted that our solution is easily *configurable*, thus other criterion different to ISP friendliness can be used (e.g. geographical criterion).

In a nutshell, we have demonstrated that our solution fulfils the 11 design criteria specified in the introduction of this section. In addition, we have compared our solution with the related work demonstrating that our solution is the one that better fulfils the defined design requirements. Table 6.1 summarize the discussion in this Section.

Next, we present two specific case of study where we use our solution to discover a close-by NAT Traversal Server and a close-by Home Agent (IP Mobility) respectively.

Solution	Infrastructure	Unexpected End-Host Probing	Robust to TIV	Robust to Path Variability	Robust to Churn	Robust to Mobility	Measurement Overhead	Maintenance Overhead	Complexity	Designed for Server Discovery	Selection based only distance/delay	ISP Friendliness
IDMaps	YES	YES	NO	NO	YES	YES	High	High	Medium	NO	YES	NO
Dynamics Maps	YES	YES	NO	NO	YES	YES	High	High	Medium	NO	YES	NO
iPlane	YES	YES	NO	NO	YES	YES	High	High	High	NO	YES	NO
Cent. NCSs	YES	NO	NO	NO	PARTIALLY*	NO	High	High	High	NO	YES	NO
Vivaldi	NO	NO	NO	NO	PARTIALLY*	NO	Medium	Medium	High	NO	YES	NO
Pyxida	NO	NO	NO	NO	PARTIALLY*	NO	Low	Medium	High	NO	YES	NO
Htrae	NO	NO	NO	NO	PARTIALLY*	NO	Medium	Medium	High	NO	YES	NO
Meridian	NO	YES	YES	YES	PARTIALLY*	YES	High	High	High	YES	YES	NO
OASIS	YES	YES	YES	YES	PARTIALLY*	YES	Medium	High	High	YES	NO	NO
desired	NO	NO	YES	YES	YES	YES	Very Low	Very Low	Very Low	YES	NO	YES
Our Solution	NO	NO	YES	YES	YES	YES	Medium	Low	Low	YES	NO	YES

Table 6.1: *Comparison of potential solutions for Dynamic and Location Aware Server Discovery (*PARTIALLY means that churn leads to a higher overhead).*

Chapter 7

Location Aware NAT Traversal Server Discovery

In this chapter we apply the proposed solution to the discovery of NAT Traversal Servers. We detail the problem and the proposed architecture, named Peer-to-Peer Nat Traversal Architecture (P2P-NTA). The solution relies in a Sequential Algorithm named Gradual Proximity Algorithm (GPA) as core of the Server Discovery Procedure. We validate this solution with real data obtained from the iPlane project [iPl].

It is worth noting that this chapter of the Thesis has been published in [CCCA+10].

7.1 Overview

In the current Internet picture more than 70% of the hosts are located behind Network Address Translators (NATs). This is not a problem for the client/server paradigm. However, the Internet has evolved, and nowadays the largest portion of the traffic is due to Peer-to-Peer (P2P) applications. This scenario presents an important challenge: two hosts behind NATs (NATed hosts) cannot establish direct communications. The easiest way to solve this problem is by using a third entity, called Relay, that forwards the traffic between the NATed hosts. Although many efforts have been devoted to avoid the use of Relays, they are still needed in many situations. Hence, the selection of a *suitable* Relay becomes critical to many P2P applications. In this chapter we propose the *Gradual Proximity Algorithm (GPA)*: a simple algorithm that guarantees the selection of a topologically close-by Relay. We present a measurement-based analysis, showing that the GPA minimizes both the delay of the relayed communication and the transit traffic generated by the Relay, being a QoS-aware and ISP-friendly solution. Furthermore, we present the Peer-to-Peer NAT Traversal Architecture (P2P-NTA), which is an specific version of the solution described in Chapter 5. This architecture addresses the Relay discovery/selection problem. We have performed large-scale simulations based on real measurements, which validate our proposal. The results demon-

strate that the P2P-NTA performs similarly to direct communications with reasonably large deployments of P2P applications. In fact, only 5% of the communications experience an extra delay that may degrade the QoS due to the use of Relays. Furthermore, the amount of extra transit traffic generated is only 6%.

7.2 Introduction

The rapid reduction of the IPv4 address space [VB07] has stimulated the widespread deployment of Network Address Translators (NATs) [EF94] on the Internet. Furthermore, companies' network security policies include the utilization of NATs and/or firewalls in order to hide the network topology and control both inbound and outbound traffic. As a result, recent studies state that more than 73% of Internet end-hosts are located behind NATs or firewalls [CF07]. In the following we will use the term *NATed* to designate entities behind a NAT.

NATs were designed for client/server applications. However, in the last decade Peer-to-Peer (P2P) applications such as VoIP (e.g., Skype), on-line games, P2P file sharing (e.g., BitTorrent), P2P streaming (e.g., PPLive) have become tremendously popular and nowadays they are responsible for the largest share of Internet traffic [CAC07]. Unfortunately, P2P communications cannot be directly established through NATs. This is because NATs do not allow inbound connections unless they are manually configured to do so. Researchers have designed a set of techniques to provide NAT traversal capabilities to the NATed hosts [RWHM03, RMM09, Ros07, FSK05, GF05, BFWP05]. However, there are a large number of cases (e.g. symmetric NAT) [HFC$^+$08] where these techniques do not work. In these cases the communication must be established using a third non-NATed entity that we call Relay (also known as NAT Traversal Server). In such a scenario, both end-hosts communicate through this Relay (*end-host1* $\leftrightarrow$ *Relay* $\leftrightarrow$ *end-host2*), that forwards the traffic between them.

As a consequence, the selection of an appropriate Relay becomes a key issue that has a direct impact on the communication delay and, in addition, it may avoid extra-costs for the ISP that hosts the Relay. In particular: (i) a bad choice of the Relay may increase the delay, leading to an undesirable QoS experienced in delay-sensitive communications such as multimedia (VoIP, on-line gaming, etc); (ii) if the Relay is not carefully selected, it increases the total transit traffic of the ISP that hosts it. It must be noted that some P2P applications with a large number of NATed users, such as BitTorrent or eMule, generate a large amount of traffic that may produce an increase of the ISP's costs. Therefore, it is critical to define a Relay selection algorithm that eliminates, or at least alleviates, these negative effects.

This chapter presents the *Gradual Proximity Algorithm*[1] (GPA). GPA is a Relay selection algorithm that chooses a topologically close-by Relay from the available pool: first, it tries to find a Relay in the same Autonomous System (AS) as the NATed client. In case there is no Relay in that AS, a Relay within the same NATed node's country is selected; if this fails too, the GPA selects a Relay in the NATed node's continent (to avoid transcontinental links);

[1] This is an example of the sequential search mechanism introduced in Chapter 5.

finally if all the previous attempts fail, a random Relay is chosen. We rely on real measurements to demonstrate that the GPA minimizes the delay of the relayed communication. Also, this algorithm minimizes the extra transit traffic of the Relays' ISPs.

Additionally, this paper presents the Peer-to-Peer NAT Traversal Architecture (P2P-NTA). This is a wide area collaborative solution that addresses the problem of Relay discovery and selection by using the GPA. Basically, the P2P-NTA is a distributed solution where Relays form a Distributed Hash Table (DHT) to store their location information: AS, country and continent. The NATed nodes are connected to a Relay that belongs to the DHT. In the case when the node is connected to a distant Relay (e.g., on a different continent), the former asks the latter for a closer Relay. The Relay then runs the GPA: (i) it queries the DHT to find a Relay in the same AS as the client; (ii) if there is none, it queries for a Relay in the same country; (iii) if no Relay is found in the same country, it asks for a Relay on the same continent. After that, the Relays found are sent to the NATed host that connects to the closest one with enough available bandwidth.

Therefore, if we assume that the DHT is well populated and there is at least one Relay per AS, the relayed communications would experience a latency similar to the direct counterpart. This is because we are just adding an additional intra-AS hop that is likely to have a low associated delay. Moreover, all relayed communications follow the same AS-path than the direct one, producing no extra transit costs to the ISPs.

In order to validate the P2P-NTA architecture we have developed the P2P-NTA simulator that uses the real Internet AS-topology and real end-to-end latencies. This data has been obtained from the iPlane project [MIP+06, iPl]. In particular, this project provides AS connectivity, IP prefixes announced in the default-free zone and delay measurements between pairs of Points of Presence (PoPs) in the Internet. This dataset has allowed us to run very large scale simulations involving thousands of real ISPs and several thousands of end-hosts. The results assess the validity of our proposal: its performance is similar to that of direct communication when considering a reasonably sized deployment. Less than 5% of communications suffer from an extra delay that may affect the QoS. Moreover, the P2P-NTA generates an almost negligible amount of transit traffic (6% more than the case of direct communications), thus confirming that it is an ISP friendly solution. On the other hand, the P2P-NTA clearly outperforms other Relay selection algorithms such as Random or Pre-Established Relay selection. In these proposals, more than 50% of the communications suffer from an extra delay that may affect the QoS, whereas even in the best case over 85% of extra transit traffic is generated.

In short, the main contributions of this chapter are:

- *The Gradual Proximity Algorithm*: This is a lightweight and simple algorithm that allows finding a topologically close-by Relay from the available pool. This allows to minimize the relayed communication delay and avoids to generate extra transit traffic at the Relay's ISP.
- *The Peer-to-Peer NAT Traversal Architecture (P2P-NTA)*: This is a globally distributed and collaborative architecture that solves the problem of Relay discovery and selection. For this purpose it uses a DHT to register and retrieve the location information

of the Relays, and implements the GPA discovery procedure. This lightweight architecture inherits the advantages of the GPA.

7.3 Smart Relay Selection: the Gradual Proximity Algorithm

In this section we first discuss why Relays are a must in the current Internet. Next, we clarify why selecting a topologically close-by Relay is efficient and we describe the *Gradual Proximity Algorithm (GPA)*. Finally, we present a measurement-based analysis that validates the proposed algorithm.

The need of Relays

Around 73% of Internet users are located behind NAT [CF07]. It has been shown that even using very sophisticated techniques [Ros07, FSK05, GF05, BFWP05], there are some types of NATs (e.g Symmetric NAT), widely deployed [HFC$^+$08], that can be hardly traversed. In addition, not all the applications implement these NAT traversal techniques. This leads to the conclusion that Relays are a necessity of today's Internet.

Selecting a topologically close-by Relay

When two nodes (A and B) that are connected behind a NAT want to communicate through a Relay they establish two different connections: $A \leftrightarrow Relay \leftrightarrow B$. Some previous overlay routing proposals suggested to select the Relay randomly [GMG$^+$04], or from a pre-established pool [ABKM01]. These selection algorithms may obtain an unsuitable Relay that increases the communication delay and the transit traffic of the ISP that hosts the Relay. Indeed, it is possible that two end-users located in the same ISP choose a Relay from a different one, or even from a different continent.

We claim that selecting a topologically close Relay reduces both the delay of the communications and the transit traffic of the ISP that hosts the Relay. The Internet is structured into Autonomous Systems (AS), hence the closest Relay, in terms of hops, is usually located in the same AS as the node itself. Unfortunately, not all of the ASes may host a Relay. For this case we have designed the *Gradual Proximity selection Algorithm (GPA)*, that defines different degrees of proximity in the current Internet scheme (Alg. 1). This is a particular implementation of the *Sequential Search Procedure* explained in Chapter 5 .

Next, we explain the consequences that the GPA has in terms of delay and transit traffic cost.

7.3.1 Consequences of GPA in the communication delay

The idea behind our algorithm is to use the shortest possible AS-Path between the two end hosts, the main rationale being the following: (i) if we use a Relay in the same AS, the AS-Path will be the same as for direct communication. Then, GPA adds just some extra hops inside the AS at IP level; (ii) if we choose a Relay in the same country we

Algorithm 1 Gradual Proximity Algorithm

```
/* Input Parameters*/
userAS, userCountry, userContinent
/* Output Parameters*/
Relay
/* Algorithm*/
if ∃ Relay on the same AS then
   Choose the closest one with enough bandwidth among them
else if ∃ Relay on the same country then
   Choose the closest one with enough bandwidth among them
else if ∃ Relay on the same continent then
   Choose the closest one with enough bandwidth among them
else
   Choose a random Relay with enough bandwidth
end if
return Relay
```

are probably adding one AS-hop toward the Relay[2]; (iii) if we select a Relay on the same continent, it is likely to add some hops to the AS-Path, but we avoid transcontinental links (e.g., transoceanic) that have a very high propagation delay.

To validate the performance of the proposed algorithm in terms of delay, we have performed a set of live experiments. We have deployed measurement boxes in different ASes: 3 in Spain: ES1, ES2, ES3; 3 in three different European countries: EU1 (Italy), EU2 (Norway), EU3 (Greece); 1 of them located in the US, US1; and the last one located in India, IN1. We use the notation $u1$ and $u2$ for the end users involved in the communication and R for the Relay. The location of $u1$ is fixed to ES1 while we iterate $u2$ and R among all the possible locations. Figure 7.1 shows the RTT values of the relayed communications between $u1$ (always located in ES1) and $u2$ (located in the AS indicated by x-axis) through R (located in the AS indicated by the legend)[3].

As the figure shows, the topological distance between the Relay and the end-users has a significant impact on the communication delay. For instance, IN1 has the largest topological distance from any ES or EU location. Thus, for all those cases where u1 and u2 are located anywhere in Europe, using a Relay in India produces the highest delay. We can also see that, for a given $u2$ location, the latency increases as assumed by the GPA: same AS $<$ same country $<$ same continent $<$ different continent.

To further validate our algorithm we have measured the end-to-end delay of a large set of end-users. The measurements come from iPlane [MIP$^+$06, iPl], which is a scalable service, providing accurate predictions of Internet path performance for overlay services and Internet-scale simulations. To achieve these goals, the iPlane project uses hundreds of vantage points distributed across the Internet for measurements, updating their dataset

[2]It is likely that two ISPs within the same country have a peering agreement, thus being one AS-hop apart to each other.

[3]We measured the RTT, for each end-user to end-user communication, 10 times per day at different day hours during one week. Fig 7.1 shows the average value of the RTT.

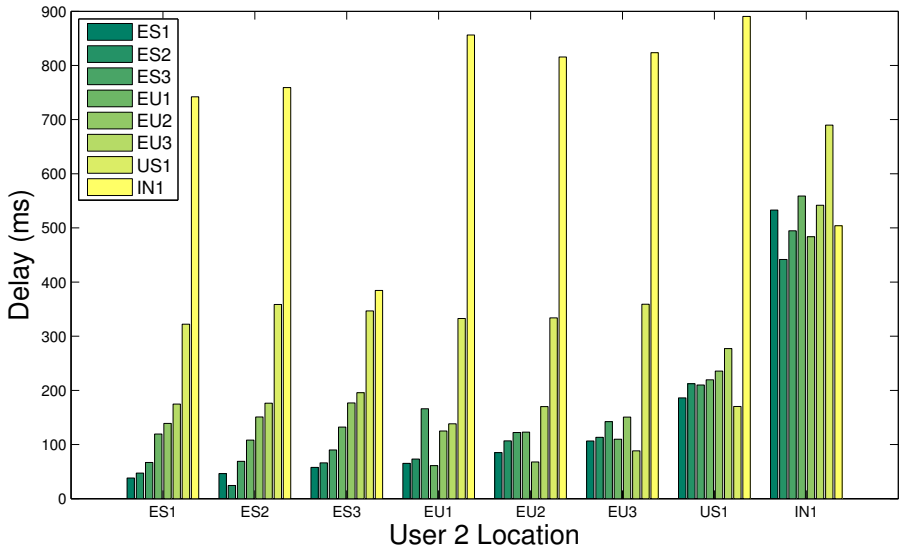


Figure 7.1: *RTT of Relayed Communication ($User\ 1 \leftrightarrow Relay \leftrightarrow User\ 2$). User 1 is always located in ES1. X-axis identifies User 2 AS. Each bar represents the Relay's AS as indicated by the legend.*

daily. iPlane is based on daily active latency measurements from various vantage points of the Internet. In particular they take advantage of the PlanetLab infrastructure and, using traceroute, they monitor hundreds of paths from each of the available PlanetLab nodes. In this experiment we have obtained a latency dataset by querying the iPlane service using random IP addresses. The iPlane interface includes in each reply a flag indicating if the requested latency has been either estimated or measured. In our dataset we only consider measured latencies.

In particular, we have measured the one-way delay for: (i) end-users located in the same AS; (ii) end-users located in the same country but different ASes; (iii) end-users located on the same continent but different country; (iv) end-users located on different continents. Our dataset contains 1M end-to-end delays. Figure 7.2 shows the Cumulative Distribution Function (CDF) of the delay for the different cases. In this context the term delay refers to the latency measured by iPlane using traceroute, and targets the instantaneous delay.

These distributions suggest that the delay is strongly related to the topological distance between the end-users. If we consider the maximum one-way delay recommended by the ITU-T for voice communications (150 ms) [itu96], we can conclude according to this experiment that more than 95% of the communications are below that threshold, if both end users belong to the same AS. This percentage is still over 90% when the users are located in the same country but in different ASes. When the users are located on the same continent, the percentage drops to 83%. Finally the delay is severely impacted (40%) if the users are on different continents, and the end-to-end path includes trans-continental links.

It is also worth to note that while ITU recommends 150ms as a quality threshold for voice

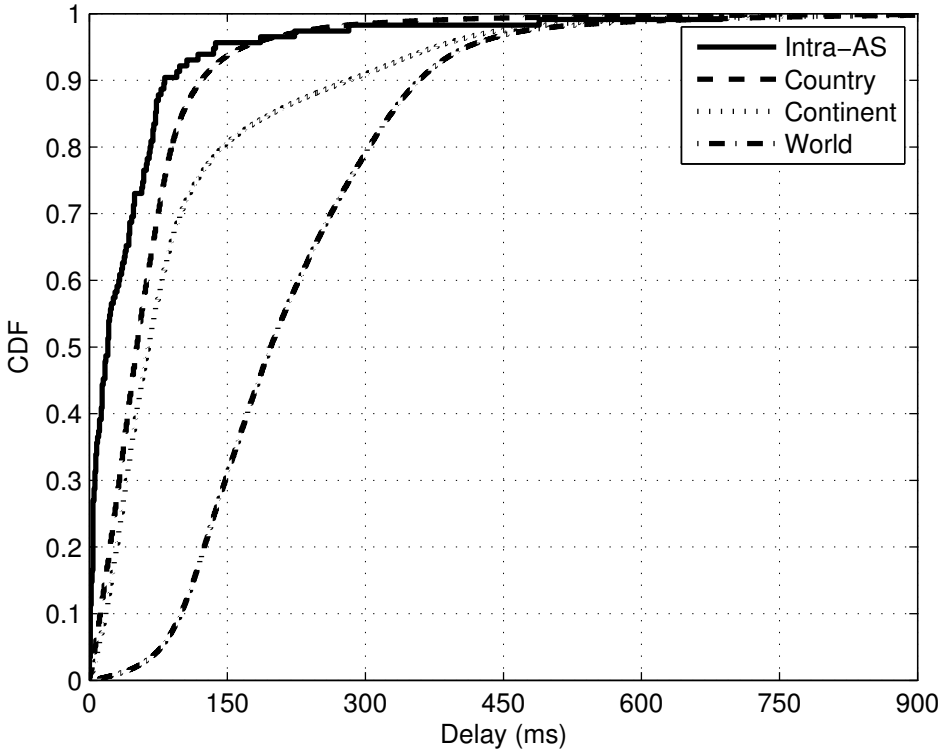


Figure 7.2: *One way Delay for different topological end-user locations*

communications, empirical experiments run by Cisco show that there is a negligible difference in voice quality Mean Opinion Score (MOS) when a 200ms threshold is used [CIS05]. Throughout this chapter we will use a 200ms delay threshold to differentiate between acceptable and non-acceptable quality voice communications. In particular, in this dataset if we consider the Cisco's threshold: 98%, 96%, 87% and 50% of the communications are below 200ms when users are located in the same AS, same country, same continent and different continents respectively.

Therefore, the obtained results confirm the suitability of GPA as Relay selection algorithm that effectively minimizes the end-to-end communication delay.

7.3.2 Consequences of GPA in transit traffic

In this subsection we evaluate the extra traffic caused by the GPA algorithm. ISPs usually pay both for inbound and outbound traffic flowing through transit links, while the cost of traffic transmitted through peering links is typically free [W.B04]. In order to better understand how the GPA avoids the transit traffic, let us focus on the case where $u1$ and $u2$ are located in different countries and ASes. In this case, *what is the extra cost that the Relay's ISP has to pay compared to the case of direct communication?*:

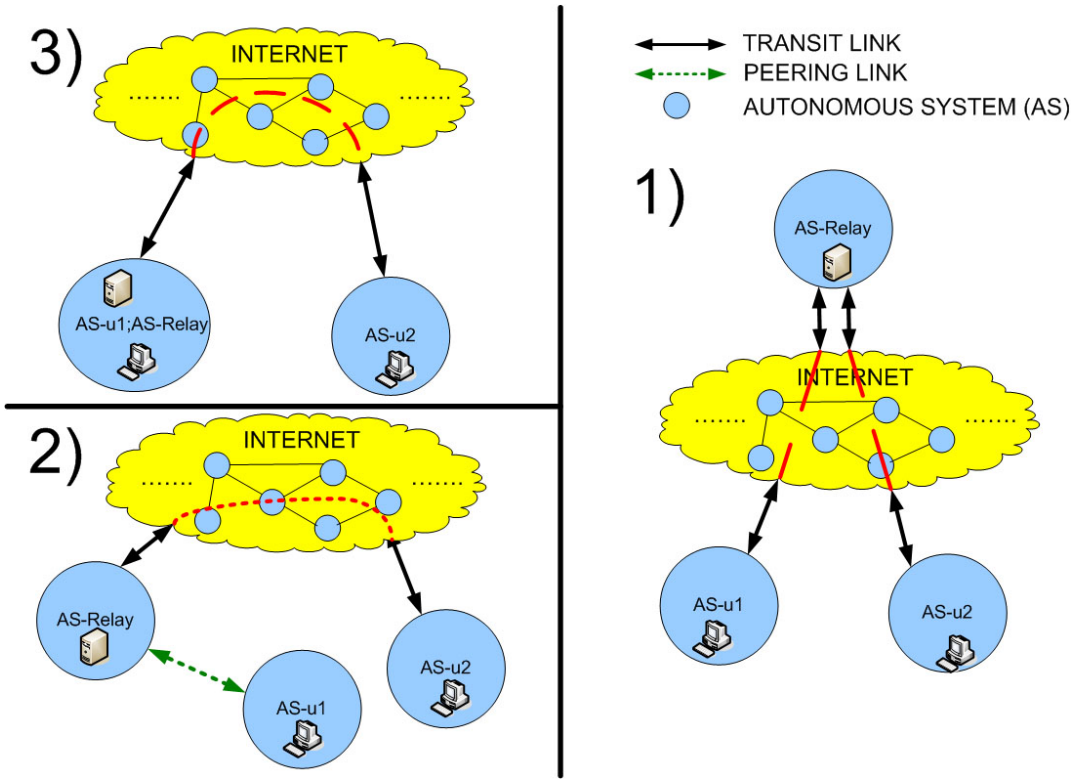


Figure 7.3: *ISP-friendly Relay Selection.* Case 1: *The selected Relay is located in a different AS and country as the users. It uses (paid) transit links to both users.* Case 2: *The selected Relay is located in an AS in the same country as one of the users. It uses a (paid) transit link to one of the users and a (free) peering link to the other.* Case 3: *The Relay is located in the same AS as one of the users. The relayed communication follows the same AS-Path as the direct communication, thus incurring no extra cost.*

1. If we choose a Relay in a country and AS other than the two end-hosts, the Relay uses transit links to communicate with both users. Therefore, the ISP where the Relay is located has to pay for the outbound and inbound traffic of $u1$ and $u2$.

2. If we select a Relay in a different AS, but in the same country as $u1$, the Relay uses a peering link to communicate with $u1$[4] whereas it uses a transit link to communicate with $u2$. Thus, the cost for the ISP where the Relay is located is half than in the previous case.

3. If we select a Relay in the same AS as either $u1$ or $u2$, the relayed AS-path is the same as the direct one, thus the ISP does not incur any extra cost.

Figure 7.3 shows the different scenarios explained above. As a result, the Gradual Proximity Algorithm always selects a Relay that minimizes the ISP transit traffic, and therefore can be considered as an ISP-friendly algorithm.

[4] Access ISPs within the same country typically establish peering agreements.

In a nutshell, we have demonstrated that our GPA leads to a reduction in the communication delays while minimizing the transit traffic costs.

Finally, it must be highlighted that these benefits could have a direct impact in currently deployed applications with millions of users. On the one hand, VoIP applications such as Skype use real-time communications that are delay-sensitive, therefore GPA would improve the quality of the communications. On the other hand, the majority of the users of P2P file sharing applications such as eMule or BitTorrent are located behind NATs[5], thus requiring a Relay. Moreover, these applications produce a large amount of traffic, and this increases the costs for ISPs. In this scenario, the GPA reduces significantly the relayed transit traffic and the costs for ISPs with regard to other proposals.

7.4 The P2P NAT Traversal Architecture (P2P-NTA)

In this section we present the *P2P Nat Traversal Architecture (P2P-NTA)*. This is a specific implementation of the solution described in Chapter 5 aiming the discovery of close-by Relay. It relies in the Gradual Proximity Algorithm as described in Section 7.3. First, we detail the proposed architecture and its functionality.

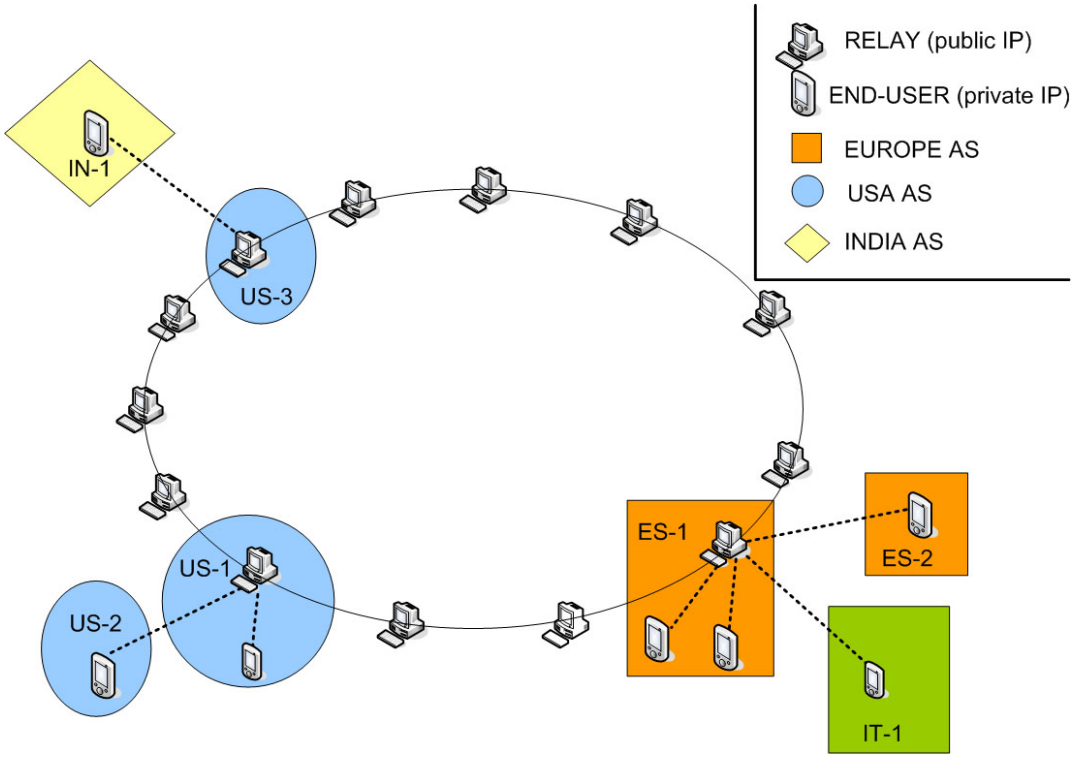


Figure 7.4: *P2P-NTA Physical Architecture*

[5]We have conducted a large-scale crawling of BitTorrent, demonstrating that more than half of the users are located behind NATs. More detailed information can be found in [KRT+09].

Physical Architecture: the *P2P-NTA* consists of two different types of nodes, *clients* and *Relays*. Nodes located behind a NAT are clients, whereas nodes having a public IP address and enough available bandwidth typically become Relays. The latter form a Distributed Hash Table (DHT) where they register their location information. Each client has an associated Relay from this DHT for NAT-Traversal capabilities. Figure 7.4 depicts the physical architecture.

Bootstrapping: When turned on, the node checks if it is a Relay (it has a public IP address) or a client (it is behind a NAT). If it is a Relay it joins the DHT. For this purpose, it computes its *Peer-ID* as $hash(IP\ address)$. This *Peer-ID* indicates the position of the node in the DHT. After that, the Relay contacts any member of the DHT (previously known, well-known peers, Bootstrapping server,...) to join the *P2P-NTA*. On the other hand, if the node is a client it needs to attach to a Relay belonging to the *P2P-NTA*. If it is the first time the node joins the P2P, it will contact a bootstrapping server or a well-known DHT peer, if not, it will connect to any Relay known in the past. After that, the client checks if this Relay belongs to its own AS and if it provides a good QoS to the node's communications (bounded delay and enough available bandwidth). In this case, the client keeps this Relay, otherwise it finds a closer one as described below (Relay Discovery Procedure).

Registration of the Relay's Location Information: Once the Relay has joined the DHT, it computes its $AS\ key = hash(ASnumber)$, $country\ key = hash(countryID)$ and $continent\ key = hash(continentID)$. Then it stores the following tuples: <AS-key, Relay IP>, <country-key, Relay IP> and <continent-key, Relay IP> in the DHT. In particular these tuples are stored on the nodes with the closest IDs to the AS-key (AS-key's responsible node), country-key (country-key's responsible node) and continent-key (continent-key's responsible node) respectively[6].

Relay Discovery Procedure (based on GPA): The client can eventually detect that its current Relay is located in another ISP, it is too distant (e.g. the RTT towards the Relay is above a predefined threshold) or it has not enough available resources (e.g. bandwidth). This triggers the discovery procedure for a closer Relay: the client sends a message to its current Relay, including its publicly visible IP address. In turn, the Relay discovers the client's AS, country and continent and with this information, it computes an *AS key*, a *country Key* and a *continent Key*. Next, the Relay applies the GPA: (i) First, it sends a query to the DHT looking for the *AS key*. If the query succeeds, the Relay obtains a list with all the IP addresses of the Relays located in the same AS as the client. (ii) If this first query fails, the Relay sends a second query looking for the *country Key*, if this query succeeds the Relay retrieves the list of IP addresses of Relays located in the same country as the client. (iii) If the query fails again, the Relay sends a third query asking for the *continent Key*. The Relay, in turn, forwards this information to the client. Thus, the client obtains a list of, either all the Relays located in its AS (if the first query succeeded), all the Relays located in its country (if the second query succeeded) or all the Relays located on its continent (if the third query succeeded). If none of the queries succeed the client keeps its current Relay (instead of selecting a random one as explained in Section 7.3). It could be possible that the number of Relays located in

[6]The *responsible node* refers to the node of the P2P that stores and is authoritative for the requested information.

a given AS, country or continent is too large (hundreds or even thousands). In this case the responsible node would not answer with the full list but rather a limited number (e.g., 50) selected at random.

Relay Selection Procedure: The client has to select a Relay from those included on the list. In order to select the best one in terms of available bandwidth and delay, it first contacts the closest one (i.e. the one offering the lowest RTT) and solicits to join. If that given Relay has enough available bandwidth, it accepts the client. Otherwise the client is rejected and tries the second closest Relay, repeating the process until it gets accepted.

Communication Establishment: After the Relay Selection Procedure each user is bound to a specific Relay. When two users ($u1$ and $u2$) want to establish a communication they use standard mechanisms, such as ICE [Ros07], to exchange information about their respective Relays ($R1$ and $R2$). With this information the users can test both paths ($u1$-$R1$-$u2$ and $u1$-$R2$-$u2$) and choose the best Relay, in terms of delay, to establish their connection.

Geolocation Procedure: In our solution, the Relays must be capable of identifying the country, continent and AS number associated to the client's IP address. For this purpose different public services [who] and databases [Max, SPR09] can be used.

Churn and Replication: The P2P-NTA users are, in fact, end users that may join and leave the system at any moment. This phenomenon is known as churn. In the P2P-NTA, when a Relay leaves the system gracefully, it notifies the necessary DHT nodes and removes the registered information contacting the nodes responsible of its AS, country and continent. Furthermore, it informs its neighbours and, if necessary, reassigns its stored information (AS, country and continent keys) to them. If the Relay leaves the system abruptly, it leaves the DHT with inconsistent information. To deal with such cases we have defined the following mechanisms:

- *Tuple timer*: The responsible node of an AS, country or continent tuple uses an expiration timer. The Relays must update the tuples before the timer expires, otherwise the responsible node removes the tuple. This way, if a Relay leaves the system abruptly the tuples related to it would automatically expire after a certain amount of time.

- *Replication*: The responsible node replicates the stored information in R replicas to the DHT. These are nodes with the i^{th} (i $\in$ [2,R+1]) immediate higher IDs to the given key. Thus, if the responsible node unexpectedly leaves the system, it is not affected at all, since the first replica takes the responsibility of its keys. In addition, the use of replicas enable load balancing mechanisms [GS05, GLS$^+$04, KR04, CUnB09] to share the load among all of them.

Finally, when a Relay leaves the system, its clients must select a new one by triggering the *Relay Selection* algorithm as described above. The worst case happens when a Relay, which is forwarding traffic between two different users, leaves the system abruptly. Since each user is bound to a given Relay, and they have agreed on using one of them during the *Communication Establishment* procedure, they can switch immediately to the other one and resume their communication.

7.5 The P2P-NTA Simulator

This section describes the P2P-NTA simulator used to validate the proposal. This is an iterative simulator, implementing a Chord DHT [SMLN+03] with data from users deployed in the real Internet topology. Moreover, it is using real latencies to account for the communication delays among the nodes.

The foundation of the simulator is the iPlane [MIP+06, iPl] platform (described in Section 7.3). It builds the topology based on this dataset, which contains AS connectivity, the IP prefixes announced in the BGP default-free zone and delay measurements between pairs of Points of Presence (PoPs) in the Internet. Each PoP, as defined by iPlane, is a set of IP addresses with low latency among them. The simulator uses the PoPs to build the Internet-topology, where we consider 55.000 PoPs and their actual point of attachment. Note that an AS may contain more than one PoP. The P2P-NTA simulator considers these PoPs as the access routers of ISPs and therefore, it deploys the Relays randomly among them. We consider 4 cases: 100, 1000, 10000 and 25000 Relays, each case referring to the amount of Relay nodes contained in the Chord DHT.

Then, for each iteration, the P2P-NTA chooses two different random users from two different PoPs. The users are chosen according to the following criteria, in order: within the same country, within the same continent or from different continents. With this set of experiments we aim to show the performance of our proposal under different scenarios.

The users query the Chord network deployed among the Relays that run the GPA selection algorithm. The P2P-NTA simulator implements a highly scalable Chord network and it is able to route the query towards its destination, and provide the path, number of hops, and latency. In order to simulate very-large Chord networks, the P2P-NTA uses a steady-state approach, and only simulates this P2P network after it has stabilized. We assume that during the simulation no churn is observed, and the finger table for each node is iteratively generated in turn, knowing a priori the full list of the nodes in the overlay. After the finger tables are generated, queries can be routed by the simulator using this topology. It is worth noting here that we validated our implementation of the Chord protocol in steady state with OpenChord 1.0.5[7]. Specifically we created a P2P network using OpenChord and waited until the network converged. Next, we compared the finger tables of the P2P-NTA simulator and OpenChord nodes, which were found identical.

Finally, once the query has finished, and both users have agreed on communicating using a given Relay with the help of the GPA, the simulator computes both the direct and the relayed delay[8]. In order to estimate these latencies, iPlane provides the delay between PoPs, but not the delay between the PoP and the end-user (i.e. the access link). This part of the end-to-end delay is estimated using the dataset provided in [Sta], that measures the median access link speed for different countries. Hence, the user is geolocated using the MaxMind

[7] `http://open-chord.sourceforge.net`

[8] We define direct delay as the latency between two nodes in the Internet that communicate using the standard inter and intra-domain routing protocols. We define relayed delay as the latency between two nodes that communicate through a third node.

database [Max][9] and the access link latency is estimated.

Preliminary experiments showed us that, for the scenarios simulated, iPlane provided the latency between two PoPs approximately in 70% of the cases. That is why we included a latency estimator for the remaining 30% cases. In order to design it we have used a dataset that contains roughly 200k latencies[10] between arbitrary pairs of hosts. We have divided this dataset (randomly) into two sets, one for training and designing the estimator, and the other one for validation purposes.

In order to design a latency estimator we take into account the information that we can associate to each PoP. In particular we aim to correlate the geographical distance between them with the latency and we consider the following estimators. First a linear regression, secondly we bin the pairs of PoPs depending on their distance and we compute the Experimental Cumulative Distribution Function (ECDF) of the latencies. Then, considering this training data, we estimate the delay of a pair of PoPs firstly computing the distance between them, and then generating a random number that follows the ECDF of the appropriate bin. In particular we consider two bin sizes: (i) (0-10km, 10-100km, 100-1000km, 1000-10.000km, 10.000-20.000km) and (ii) (0-10km, 10-100km, 100-500km, 500-1000km, 1000-2500km, 2500-5000km, 5000-7500km, 7500-10.000km, 10.000-15.000km, 15.000-20.000km). With this approach, we assume that there is a correlation between a given bin and the latency, for instance routers that are at a range of 10km may have the same amount of hops on their paths. Further, we also consider this approach taking into account the particularities of their location at a continent-level. It is clear that the topology of the Internet is different if we consider North America or Europe, mainly because some continents are more densely populated, and routers may be deployed closer.

Figure 7.5 shows the error of the estimator. Each curve represents the ECDF of the absolute error (estimated-real) of the different proposed estimators. The absolute error has been computed subtracting the real latency from the estimated one (from the validation set). As we can see the accuracy of the estimators is similar, except for the $distance/c$ estimator, which always under-estimates. This is because it only considers the propagation delay, and assumes that end-to-end paths are just a link. Regarding the rest of the estimators the linear regression is slightly more accurate than the rest of them. Further, this estimator is very fast, and will not slow down the simulator. It is important to note that generating random numbers that follow a certain ECDF is computationally intensive. Also, as Figure 7.5 shows, the linear regression estimator is not biased, and since we plan to carry out a large amount of repetitions, this will not impact the results.

7.6 Evaluation

In this section we present the obtained results from our large-scale simulations in terms of: (*i*) delay; (*ii*) ISP-friendliness; (*iii*) overhead produced by the P2P-NTA (lookup latency

[9] This database is open source and has an 99.8% accuracy at country level, 75% accuracy at city level (within a range of 25 miles), 22% accuracy at more than 25 miles, and 3% that the IP is not covered by such database.

[10] A subset of the dataset described in Section 7.3.

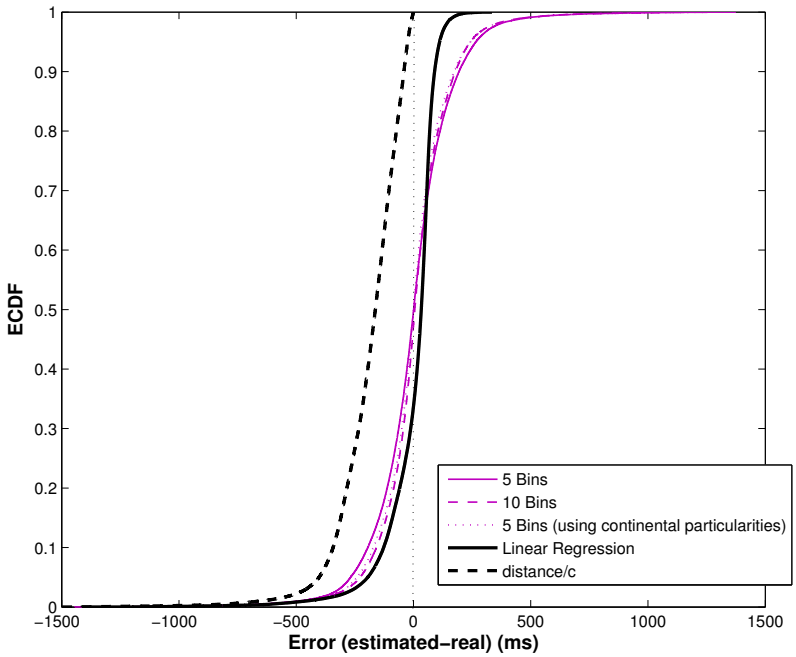


Figure 7.5: *Error of the different estimators. Each curve represents the CDF of the absolute error of the different proposed estimators.*

and Relay load).

- Simulation Set-up

We have used the P2P-NTA simulator described in section 7.5 to simulate our proposal with different deployments: 100, 1000, 10000 and 25000 Relays. These cases refer to the amount of nodes that can potentially act as Relays because they are configured using a routable IP address and are not firewalled. These nodes, which represent the P2P-NTA service, have been randomly deployed based on the 55.000 real PoPs considered by the iPlane dataset. The clients of the overlay network, using the P2P-NTA service, are also randomly deploy among the PoPs. Table 7.1 describes these scenarios in detail, showing how many ASes, countries and continents contain at least one Relay node.

We have also simulated the random Relay selection algorithm for the same number of Relays and the pre-established Relay selection algorithm for a fixed pool of 1000 Relays. These solutions are equivalent to SORS [GMG+04] and RON [ABKM01] respectively.

We have simulated, for each solution and number of Relays: (*i*) 30k communications in the *Intra-Country* scenario: the communication is established between hosts located in the same country; (*ii*) 30k communications in the *Intra-Continent* scenario: same continent (but different countries); (*iii*) 30k communications in the *Inter-Continent* scenario: different continents. In total we have simulated roughly 700k communications to evaluate our solution.

	Clients Loc. all Scenarios	Relays Loc. 100 Relays Scenario	Relays Loc. 1K Relays Scenario	Relays Loc. 10K Relays Scenario	Relays Loc. 25K Relays Scenario
#AS	15815	84	820	4996	10361
#Countries	203	26	92	149	190
#Continents	6	6	6	6	6

Table 7.1: *Distribution of Clients and Relays through ASes, Countries and Continents in the different scenarios (100, 1K, 10K and 25K Relays). Note that the Maxmind database considers that south and north America are different continents.*

For each communication we calculate the direct and the relayed delay. Also, we geolocate (AS, country and continent) the two users ($u1$ and $u2$) and the Relay (R) involved in the communication in order to estimate the *ISP-friendliness* for the different solutions. In addition, we calculate the load supported by each Relay in terms of number of relayed communications. Finally, we compute the Relay look-up latency for each communication.

- Communication Delay

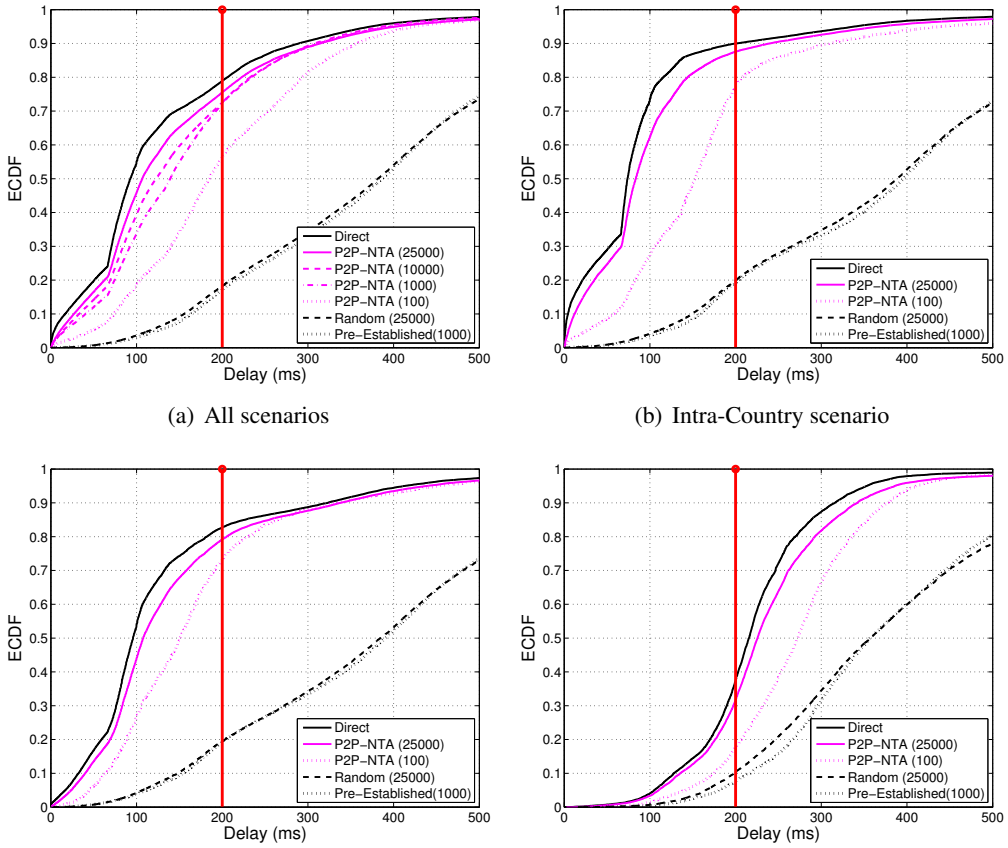
We have computed the ECDF of the one-way delay for each solution (P2P-NTA, Random Relay Selection and Pre-Established Relay Selection), deployment (100, 1000, 10000 and 25000 Relays) and scenario (Intra-Country, Intra-Continent and Inter-Continent). Figure 7.6 summarizes the obtained results:

- Figure 7.6(a) shows the ECDF for the delay for the 90k communications. We consider the Cisco's 200 ms quality threshold described in [CIS05]. That is, we consider that 200 ms is the maximum tolerable one-way delay for voice communications with acceptable QoS. As expected, the direct communications present the lowest delay, and 79% of them are below the aforementioned threshold. The P2P-NTA slightly increases the direct communication delay under the largest considered deployment. For instance, in the case of 25000 Relays[11], 75.5% of the communications are below the 200 ms threshold. This means that less than 4% of the communications would suffer from QoS degradation due to the use of Relays compared to the direct one. As the deployment decreases, the number of communications below the threshold slowly decreases (72.6% for 10000 Relays and 72.2% for 1000 Relays) up to 56.3% in the 100 Relays case. However, even in such small deployments, the P2P-NTA clearly outperforms other proposed algorithms. The figure shows that Random[12] and Pre-Established selection algorithms can only keep the communication delay below the 200 ms threshold in 18.3% and 17.2% of the cases respectively. These values are less than $1/2$ of our proposal's worst case (100 Relays).

- Figures 7.6(b), 7.6(c), 7.6(d), depict the result for the Intra-Country, Intra-Continent and Inter-Continent scenarios respectively. For clarity we just plot the deployments

[11] It must be highlighted that 25000 Relays is actually a small deployment if we consider current P2P applications such as Skype or KAD that include millions of concurrent users.

[12] Note that the random selection algorithm performs similarly regardless of the number of Relays. Thus, we only depict one case.



(a) All scenarios

(b) Intra-Country scenario

(c) Intra-Continent scenario

(d) Inter-Continent scenario

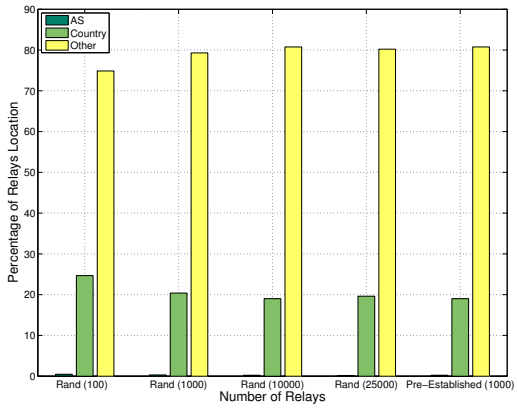
Figure 7.6: *Relayed Communications Delay*

of 25000 and 100 Relays for the P2P-NTA solution. The rest of deployments lay between these two curves. In those cases, the amount of communications that are below the quality threshold between the direct and the P2P-NTA (25000 Relays) is always smaller than 6%. Hence, independently of the type of communication considered (short, medium or long distance) we can conclude that our solution causes a minimum QoS degradation.

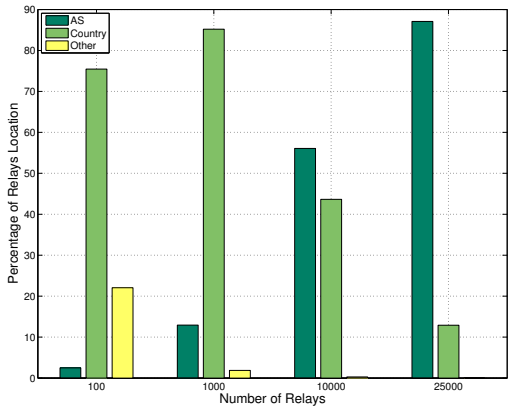
Moreover, the P2P-NTA outperforms other Relay selection algorithms. Even if we compare our solution using the smallest deployment (100 Relays), the number of communications below 200 ms is 4 times larger in the Intra-Country and Intra-Continent case, and almost 2 times larger in the Inter-Continent case.

Finally, and as a side-result, it is worth to note the problems that long distance delay-sensitive communications (e.g. VoIP) may experience in the current Internet. Figure 7.6(d) shows that even in the direct communication case, just $1/3$ of the communications are below the 200 ms threshold.

In short, we can conclude that in terms of end-to-end communication delay our solution



(a) Random and Pre-established Relay Selection



(b) P2P-NTA

Figure 7.7: *ISP friendliness. Percentage of selected Relays located in the same AS, same country or different country as the users for the different selection mechanisms*

largely outperforms other selection algorithms, and for (P2P applications) reasonably sized deployments the performance is similar to the direct communications.

- ISP-Friendliness

As stated above, ISPs have recently shown their concerns because of the large amount of traffic generated by P2P applications (e.g., BitTorrent). Figure 7.7 supports our initial hypothesis, and shows that the P2P-NTA is ISP-friendly since it minimizes the relayed traffic transmitted through transit links. This figure represents the percentage of communications (out of the 90k) in which the Relay has been selected in: (A) the AS of one of the two end-users, thus leading to zero cost; (B) an AS in the same country of one of the two hosts, thus using a free peering link to communicate to one user and a paid transit link to communicate to the other; (C) any other case where the Relay uses transit links to communicate with both end-hosts.

In Figure 7.7(a) we present the results for Random and Pre-Established selection algorithms, whereas figure 7.7(b) depicts the results for the different deployments of our solution. We can see that both, Random and Pre-Established selection algorithms, (in the best case) use less than 1% and 25% of Relays located in the same AS (case A) and country (case B) than $u1$ or $u2$, respectively. Thus, using in more than 73% of the cases transit links towards both $u1$ and $u2$ (case C).

On the other hand, and as expected, the performance of our solution is affected by the deployment. In the case of 25000 Relays, 87% of the communications are established through a Relay located in the same AS as one of the hosts (case A) while the remaining 13% use a Relay in the same country (case B). If we consider the minimum deployment case with just 100 Relays, our solution still performs quite well since just around the 20% of connections are established through a Relay outside the ISP and country of both hosts (case C).

To further understand the results, we consider relative extra transit costs compared to the case of a direct communication: $cost = 0$ for case A; $cost = 1/2$ for case B; $cost = 1$ for case C. Recall that in case B, the Relay uses a transit link to communicate with one of the hosts, whereas in case C it uses transit links for both hosts. Furthermore, case A is free of cost since the Relay is located in the same AS as one of the end-users. Table 7.2 shows the average cost of the communications for each solution. We observe that Pre-Established and Random selection algorithms are close to the maximum cost (between the 87-90% of the maximum possible cost), whereas the P2P-NTA produces almost no transit traffic cost (6%) for the largest considered deployment. The cost increases as we reduce the deployment. However, even in the case of minimum deployment (100 Relays), our solution reduces around 30% of the transit traffic compared to the other proposals.

	100 Relays	1000 Relays	10000 Relays	25000 Relays
P2P-NTA	0.60	0.44	0.22	0.06
Random Selection	0.87	0.89	0.90	0.90
Pre-Established Selection	-	0.89	-	-

Table 7.2: *Average Transit Traffic Cost of the Communications (Max = 1, Min = 0)*

P2P-NTA Overhead

In this section we evaluate the different aspects of the overhead introduced by our solution. First, we evaluate the Relay lookup latency. Next, we evaluate the number of communications transmitted through the Relay nodes, and we compare them to a Random selection approach.

- *Relay lookup latency*: We define the lookup latency as the time to search through the P2P-NTA and retrieve a list of Relays. We have measured the Relay lookup latency for all the communications (90k) and the different deployments. Figure 7.8 summarizes the results. It shows the ECDF of the Relay lookup latency for the different deployments. As we can observe, the lookup latency is in the order of a few seconds and decreases as we augment the number of Relays. Although this could seem a high

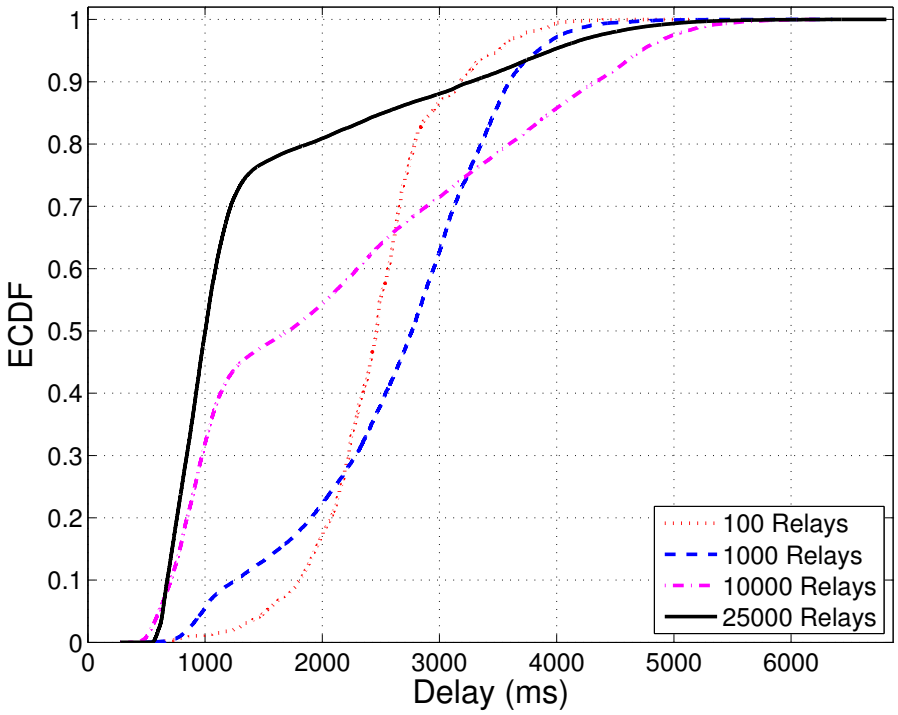


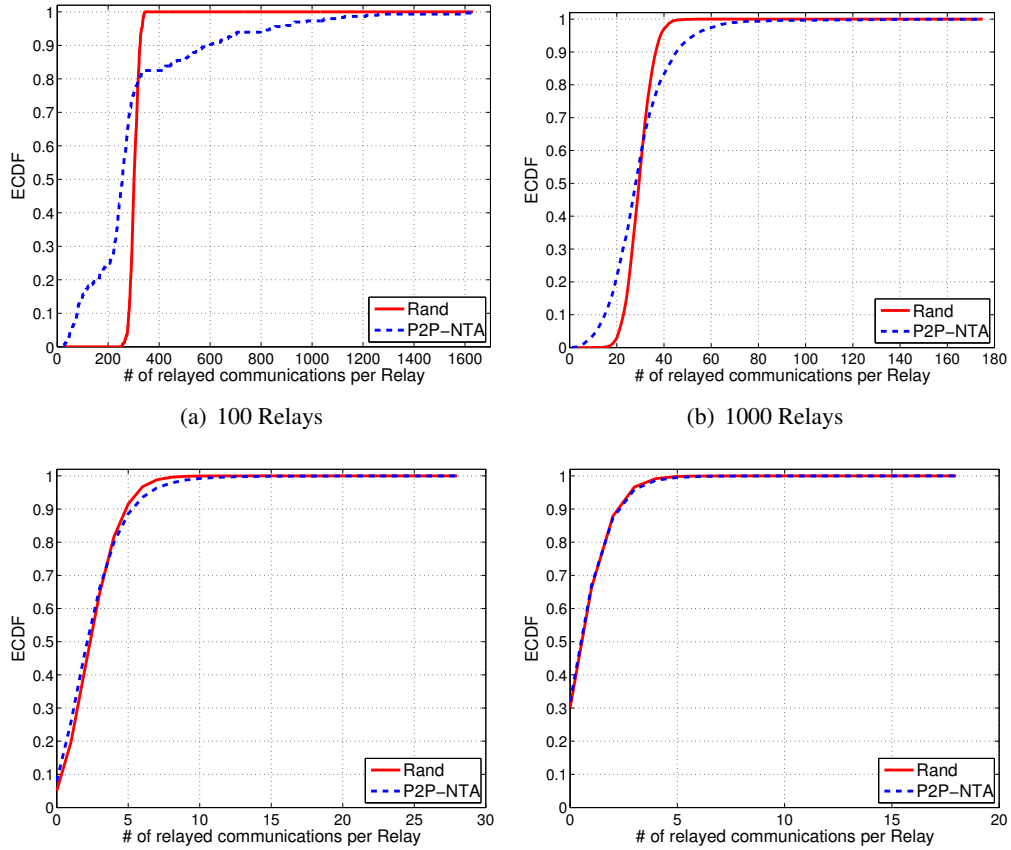
Figure 7.8: *Relay Look-up Latency for different deployments*

value, it is not affecting the QoS of the communications. Note that the P2P-NTA launches the lookup procedure when the application (e.g. Skype) starts. Thus, when the user desires to establish a communication the Relay has been already selected. Then, this does not add any extra delay to the standard connection establishment protocol (e.g., [Ros07]) in our solution.

- *Number of Supported Communications per Relay*: We have computed the number of communications supported by each Relay, for both solutions, P2P-NTA and random Relay selection, considering the different deployments. The resultant ECDFs are presented in Figure 7.9. First, we observe that the load produced by our system is similar to that generated by a random Relay selection algorithm. Since a random selection is expected to produce a fair distribution, we can claim that the P2P-NTA is a fair solution in terms of Relay usage. Actually, for large deployments (25000) the curves are completely overlapped. Hence, the higher the deployment, the fairer the solution is, while still keeping lightweight.

7.7 Related Work

Graph Representation and Network Coordinate Systems (NCSs): The networking research community has dedicated some effort to predict the Internet graph. These works give an estimation of the distance between hosts in the Internet. A first approach consists



(a) 100 Relays (b) 1000 Relays

(c) 10000 Relays (d) 25000 Relays

Figure 7.9: *Number of relayed communications per Relay for different deployments*

on creating a real Internet graph. Two of the main representatives of this approach are IDMaps [FJJ+01] and iPlane [MIP+06]. The former uses an infrastructure formed by a set of Vantage Points distributed around the Internet, named *Tracers*. The tracers measure the distance among them. The rest of the hosts are clustered in reachable Address Prefixes (APs). Furthermore, the system measures the distance between each AP and its nearest tracer. Hence, the distance between two APs can be calculated as the sum of the distance from each AP to its nearest tracer plus the distance between the tracers. iPlane [MIP+06] is based on a similar concept. However, the system implements sophisticated measurement techniques to estimate the delay, loss rate, capacity and bandwidth of the path between two Internet end-hosts. These systems need a dedicated measurement infrastructure and incur a high probing overhead.

On the other hand, the NCSs do not need an infrastructure of Vantage Points to perform measurements. In this approach, the probing is performed by all the nodes involved in the system. The aim of the NCSs is to map the system (e.g. Internet) topology into a multi-dimensional coordinate system where each node has associated a given virtual coordinate. For this purpose, each node performs measurements to other nodes in the systems in order to

find its correct position. Based on the virtual coordinate each node can estimate the distance to any other node in the virtual coordinate space. Vivaldi [DCKM04] is the most known representative of this family. More recent works have improved Vivaldi and applied the NCSs to the Azureus DHT [LGS07] and on-line games applications [AL09]. Although the NCSs do not need a dedicated measurement infrastructure they still cause a high probing load, furthermore they are not robust against the Triangular Inequality Violation (TIV) and tend to fall in suboptimal local minimum states.

These solutions help to identify the location of a given node as well as to identify the distance between nodes. Therefore, it would be feasible to apply them to the problem of Relay Selection. However, they are more complex and cause a higher probing load than our proposed solution. Additionally, they are ISP-unaware, thus they must be redesigned in order to be ISP-friendly.

Overlay Routing: It is commonly accepted by the research community that with the current AS-based routing (BGP [RLH06a]), the direct route between two end-hosts may be suboptimal in terms of delay. Hence, several solutions have been proposed, that may reduce this delay in some cases, using an overlay routing approach. These solutions use one or multiple intermediate overlay relay nodes in order to shorten this AS-path. Solutions such as RON (where the Relays are selected from a static pool) [ABKM01] or SORS (where the Relays are selected at random) [GMG+04] are intended to improve general IP routing. Other solutions such as ASAP [RGZ06] are designed only for VoIP applications. In particular, ASAP is a complex architecture to find Relays that, according to their results, reduces the delay in some cases. However, it is not clear what the signalling overhead produced by the proposed system is, and it has the disadvantage of relying on cluster representatives (called surrogate nodes) that are single points of failure. Unlike the P2P-NTA, none of these solutions consider the NAT Traversal problem. In addition, these solutions produce extra transit traffic at the ISPs where the Relays are located, imposing extra costs to these companies.

Close-by Server Selection: J. Guton *et al.* presented an early work on static and centralized location of nearby servers of a distributed service [GS95]. The solution combined traceroutes and hop-count measurements to determine the closest replica. One year latter, R. Carter et al. [CC97] demonstrated that dynamic server selection is more efficient than static server selection due to the variability of route latency over time and the large divergence between hopcount and latency. In parallel, IP Anycast was proposed as a network-layer solution to server selection. It was first proposed in 1993 by the IETF RFC 1546. However, due to various deployment and scalability problems [KW00], it has not been widely deployed. More recent solutions, Meridian [WSS05] and OASIS [FLM06], also address this problem. In Meridian, the servers form an Overlay where each server knows some other servers and locate them in concentric rings based on the measured RTT. When a given client launches a query to find a close-by server, the query progresses trough the overlay until it reaches the closest server. During the process a large number of servers have to measure the RTT to the client, what constitutes a high probing load. On the other hand, OASIS is an anycast infrastructure issued for multiple services. It has a central infrastructure of nodes used to locate a close-by server to a given client. The delay measurements are performed by the replica servers of the different services registered in OASIS. The measurement procedure is based

on an optimization of Meridian.

All the described solutions are designed for classical services where the server is expected to be an *always on-line* machine. Although they could be applied to P2P applications, the probing load would increase dramatically due to the users churn. Furthermore in the case of OASIS a dedicated infrastructure of core nodes is needed. Again it is worth noting that none of these solutions have been designed either for ISP-friendliness or for dealing with NATed hosts.

Relay/Super Node Selection: To the best of the authors' knowledge there are very few proposals that address the problem of Relay selection in case of hosts behind NATs. The P2PSIP Working Group (WG) [p2p] of the IETF is currently designing a P2P version of the SIP framework of protocols, where the users are able to establish communications among them without using any rendezvous server such as SIP Proxies or SIP Registrars. For this purpose, they use a DHT. One of their major challenges is how to solve the problem of NAT Traversal. In [Xin07], the authors propose a lightweight mechanism to discover Relays within the P2PSIP DHT. Although the protocol is promising, it does not consider any kind of location information for selecting a Relay, and this incurs the costs already mentioned in this paper. We believe that the P2P-NTA is a good candidate solution to be considered by the P2PSIP WG to solve the problem of NAT-Traversal server discovery.

Authors in [WD05] propose VIP, a P2P communication platform for NAT Traversal. In their solution, the nodes use ICE [Ros07] and Hole Punching [FSK05, GF05, BFWP05] in order to traverse the NATs. Basically, the nodes learn their available IP addresses/ports and register them into a DHT, so their *buddies* can easily access this information. Some of these VIP nodes act as Relays for those which are behind NATs. Unfortunately, the paper does not specify how a VIP node discovers one of these Relays.

Finally, some P2P applications such as Skype select as *super nodes* those that show stability and have enough available bandwidth. These *super nodes* act as Relays for that specific application. Skype is a proprietary application, and it is unknown how the Relay is selected. Nevertheless, some researchers have reverse engineered it [RGZ06, BS06, WBS08] and have found that Skype uses Relays even if direct communications are possible. In particular the clients try to establish the connection through different Relays (sometimes dozens of them) before selecting one. It is also known that the Relays are not randomly selected and that the selection is AS-unaware [RGZ06, WBS08].

Locality Solutions for P2P Applications: Over the last years ISPs are experiencing an extra transit traffic due to P2P applications. Furthermore, some ISPs have started to throttle traffic from some applications such as BitTorrent [DMHG08, vuz]. As a reaction to this problem, recently some works have appeared describing locality solutions to keep the traffic of P2P applications within the local ISP as much as possible [XYK+08, CB08]. To the best of our knowledge, there is no previous work that addresses this issue for relayed communications. Thus, we believe that our proposal is the first contribution regarding transit traffic reduction for relayed communications in the Internet.

We refer the reader to section 3.1 for a more detailed description of the related work.

7.8 Conclusions

Our work starts from the premise that relayed communications cannot be avoided in today's Internet. Based on real measurements we have shown that using a topologically close-by Relay has a critical impact on the QoS of the communications and the costs of ISPs. In order to reduce this impact we have introduced: (*i*) the Gradual Proximity Algorithm (GPA), which finds a topologically close-by available Relay and (*ii*) the Peer-to-Peer NAT Traversal Architecture (P2P-NTA), which is a lightweight distributed architecture – based on a DHT – that implements the GPA. We have carried out large-scale simulations using the real Internet topology associated with real delays. The obtained results show that our proposal exhibits performance levels comparable to direct communication and introduces a reduced amount of transit traffic when used by a reasonably large deployment of a P2P application.

Chapter 8

Location Aware Home Agent Discovery

In this chapter we apply our solution to the Location of Home Agents. We present the problem to be addressed in the case of IP Mobility and detail the solution presented in Chapter 5 for this specific case, named flexible Peer to Peer Home Agent Network (fP2P-HN).

A detailed description of different aspects of this Chapter of the Thesis can be found at [CCA+09, CACDP+09, CGC+09, CCUnG07, CCG+07].

8.1 Overview

Wireless technologies are rapidly evolving and the users are demanding the possibility of changing their point of attachment to the Internet (i.e. Access Routers) without breaking the IP communications. This can be achieved by using Mobile IP or NEMO. However, mobile clients must forward their data packets through its Home Agent (HA) to communicate with its peers. This sub-optimal route (lack of route optimization) considerably reduces the communications performance, increases the delay and the infrastructure load. In this section we present the flexible Peer-to-Peer Home Agent Network (fP2P-HN). This is a specific implementation of our solution devoted to the discovery of close-by HAs. In more detail, a Mobile Node (MN) or a Mobile Network (NEMO) can select a close-by HA to its topological position in order to reduce the delay of the paths towards its peers. Additionally, it uses flexible HAs that significantly reduce the amount of packets processed by the HA itself. The main advantages of the fP2P-HN over the existing solutions are that it is scalable and it reduces the communications delay and the load at the HAs. Since one of the main concerns in mobility is security, our solution provides authentication between the HAs and the MNs. We evaluate the performance of the fP2P-HN by simulation. Our results show that the fP2P-HN is scalable since the signalling messages overhead per HA is low and it does not increase, even if the number of deployed HAs increases. We also show that the average reduction of the communication's delay compared to Mobile IP/NEMO is at least 23% (with a minimum

deployment) and the reduction of the load at the HA is at least 54%.

8.2 Introduction

Wireless technologies have rapidly evolved in recent years. IEEE 802.11 is one of the most used wireless technologies and it provides up to 54Mbps of bandwidth in an easy and affordable way. In the current Internet status a user can be connected through a wireless link but it cannot move (i.e. change its access router) without breaking the IP communications. That is why IETF designed Mobile IP [Per02], which provides mobility to the Internet. With "mobility", a user can move and change its point of attachment to the Internet without losing its network connections.

In Mobile IP a Mobile Node (MN) has two IP addresses. The first one identifies the MN's identity (Home Address, HoA) while the second one identifies the MN's current location (Care-of Address, CoA). The MN is always reachable through its HoA while it changes its CoA according to its movements. A special entity called Home Agent (HA), placed at the MN's home network maintains bindings between the MN's HoA and CoA addresses.

The main limitation of Mobile IP is that communications between the MN and its peers are routed through the HA. Unfortunately, packets routed through the HA follow a sub-optimal path. This reduces considerably the communications' performance, increases the delay and the infrastructure load. In addition, since a single HA may be serving several MNs and forwarding several connections, the HA itself may become the bottleneck of the whole system and represents a single point of failure in Mobile IP-based networks [NTWZ07].

Mobile IPv6 [JPA04] solves this limitation by allowing MNs to communicate with their peers directly (route optimization) by exploiting special IPv6 extension headers. However, the Mobile IPv4 [Per02] and the NEMO protocol (NEMOv4 [LDNP08] and NEMOv6 [DWPT05]), which provides mobility to networks instead of nodes, do not support route optimization, even in IPv6. That is why we believe that route optimization is an issue in the current Internet status (IPv4) and even in the future (IPv6).

Solving the route optimization problem has attracted the attention of the research community and several solutions have been proposed [WOM05] [YHC06] [boe] [BGMBA06]. The main idea behind these proposals is deploying multiple HAs in different Autonomous Systems (ASes). Then, an MN may pick the best HA according to its topological position thus, reducing the delay of the paths towards its peers. The main challenge of this approach is signalling the location of the different HAs throughout the Internet. Some authors use the exterior Border Gateway Protocol (eBGP) protocol [WOM05] [boe] [BGMBA06] while others [YHC06] use Anycast routing. The main issue of these proposals is the scalability. On the one hand, using the exterior BGP protocol means increasing the load in the already oversized global routing table [Hus01]. On the other hand, anycast's defiance of hierarchical aggregation makes the service hard to scale [KW00]. In addition, these solutions force the MNs to send the data packets through the HAs, increasing the load on these devices that may become the bottleneck of the whole system [NTWZ07].

In this chapter we propose a scalable architecture, named fP2P-HN (flexible P2P Home agent Network) that solves the route optimization issue for Mobile IP and NEMO. The fP2P-HN is a simple implementation of the solution described in Chapter 5. When an MN detects that its current HA is too distant it queries its *Original HA* (the one serving the MN's Home Network) that belongs to the fP2P-HN network for a close-by HA. Then, the fP2P-HN network uses BGP information to locate a HA that reduces the delay of the paths between the MN and its peers, for instance by choosing a HA located in the same AS as the MN. Since security is one of the main concerns in mobility, we also present an architecture that provides trustworthiness to the HAs belonging to the DHT and allows that the MNs can be authenticated by the HAs (and vice versa).

Our solution allows deploying multiple HAs at different ASes without impacting the exterior BGP global routing table or requiring anycast routing; however the HAs are still responsible of forwarding all the MN's data packets. In order to alleviate their load we propose to deploy flexible HAs (fHA) [CADP07]. The main idea behind the fHAs is that a registration from an MN to a HA can be viewed as an internal route from the network's point of view. That is, when an MN registers a new location into its HA, it is actually installing a new route (Home Address → Care-of Address). This route is announced throughout the network using the interior BGP (IBGP [RLH06b]) protocol to each of the AS Border Routers. Then, the Border Routers are aware of the current location of the MN and will de-capsulate and forward any packets addressed to/from the MN directly, just as regular packets. Thus, MN's data packets are not forwarded by the HAs but by the Border Routers. It is worth to note that HAs are not necessarily devices designed for routing purpose whereas routers are routing-dedicated devices.

Our solution is simple, scalable and secure. Moreover it does not require deploying any new entities on the Internet. At the Inter-domain level, we signal the location of the HA using a DHT instead of using eBGP or anycast. At the Intra-domain level we signal the location of the MN using IBGP, in this way the Border Routers are aware of the location of the MN and the load of the HA is significantly reduced. As we will see later, we evaluate the performance of our proposal through simulation. Our results show that the fP2P-HN is scalable since the signalling overhead per HA is low (around 20kbps per HA in the worst case) and does not increase, even if the number of deployed HA increases. We also show that the average reduction of the communication's delay compared to Mobile IP/NEMO grows from 23% (with a minimum deployment) up to 80% (with large deployments). Whereas the reduction of the load at the HA varies between 54% (in the worst case) to nearly 100% (in the best case).

8.3 IP Mobility Background

8.3.1 Network Mobility (NEMO)

The IETF standardized a solution, called NEMO Basic Support Protocol (specified in the RFC 3963 [DWPT05] for IPv6 and in the RFC 5177 [LDNP08] for IPv4), to provide network mobility support. This solution defines a mobile network (known also as Network

that Moves - NEMO[1]) as a network whose attachment point to the Internet varies with time. The router within the NEMO that connects to the Internet is called the Mobile Router (MR). It is assumed that the NEMO has a Home Network where it resides when it is not moving. Since the NEMO is part of the Home Network, the Mobile Network has configured addresses belonging to one or more address blocks assigned to the Home Network: the Mobile Network Prefixes (MNPs). These addresses remain assigned to the NEMO when it is away from home. Of course, these addresses only have topological meaning when the NEMO is at home. When the NEMO is away from home, packets addressed to the Mobile Network Nodes (MNNs) will still be routed to the Home Network. Additionally, when the NEMO is away from home, i.e. it is in a visited network, the MR acquires an address from the visited network, called the Care-of Address (CoA), where the routing architecture can deliver packets without additional mechanisms. The goal of the network mobility support mechanisms is to preserve established communications between the MNNs and external Correspondent Nodes (CNs) despite movement. Packets of such communications will be addressed to the MNNs addresses, which belong to the MNP, so additional mechanisms to forward packets between the Home Network and the NEMO are needed. The basic solution for network mobility support essentially creates a bi-directional tunnel between a special node located in the Home Network of the NEMO (the Home Agent), and the Care-of Address of the MR. Fig. 8.1 shows the basic operation of the NEMO Basic Support Protocol. Inefficient routing due to the triangular CN-HA-MR path could introduce delay not desirable in some kind of applications (e.g. real time applications).

8.3.2 Terminal Mobility (Mobile IP)

The mobility support for terminals is addressed by the IETF in the RFC 3344 [Per02] for IPv4 and in RFC [JPA04] for IPv6. This can be viewed as a simplified example of Network Mobility in which the mobile entity is a single device rather than a complete network. This single node is named Mobile Node (MN) and has a Home Network (HN). The MN has configured an IP address from its HN named Home Address (HoA). This address remains assigned to the MN when it is away from home. When the MN moves, it is attached to the Internet from a Visited Network (different than the Home Network) from which it receives an IP address called Care of Address (CoA). As occurred in NEMO, the goal of the terminal mobility support mechanisms is to preserve established communications between the MN and external Correspondent Nodes (CNs) despite movement. For this purpose, the packets belonging to these communications are addressed to the MN's HoA. The basic solution is similar to that one defined for Network Mobility: a bi-directional tunnel between a special node located in the Home Network of the MN (the Home Agent) and the Care-of Address of the MN is created. Therefore, the terminal mobility also face the routing inefficiencies produced by the triangular path CN-HA-MN.

[1] NEMO can mean NEtwork MObility or NEtwork that MOves according to the context.

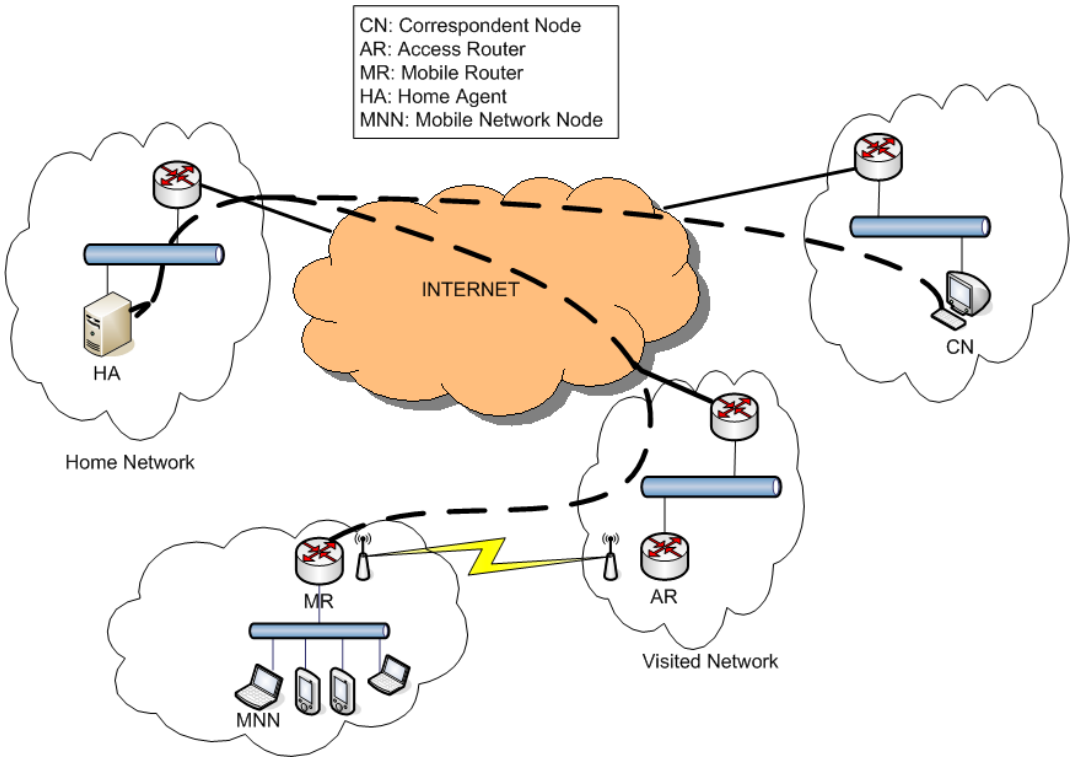


Figure 8.1: *NEMO Basic Support Protocol operation*

8.4 Flexible P2P Home Agent Network

In this section we detail the fP2P-HN architecture. This is a simple implementation of the solution proposed in Chapter 5 that specifically aims the discovery of close-by Home Agents. Please note that an fHA (flexible HA) is a Home Agent that belongs to the architecture and that has special features. We will refer to a HA or an fHA indistinctively.

8.4.1 Overview

The main goals of the fP2P-HN architecture are to reduce the delay of the communications of the MNs and the load at the fHAs. Figure 8.2 shows an overview of the architecture.

When a Mobile IP or NEMO client changes its point of attachment to the Internet it establishes a new tunnel with its HA to communicate. Depending on the MN's topological position, this new path may have a large delay. We propose to deploy several HAs throughout the Internet in order to reduce this delay. When the MN detects that the new path to its currently assigned HA has an unacceptable performance (e.g. RTT $\geq$ a given threshold) it queries its *Original* HA (the HA at the MN's administrative domain) for a close-by one (i.e. an HA located in the MN's current AS). Our architecture is flexible and allows using any metric to trigger the discovery of a close-by HA. In this work we use the RTT because it is

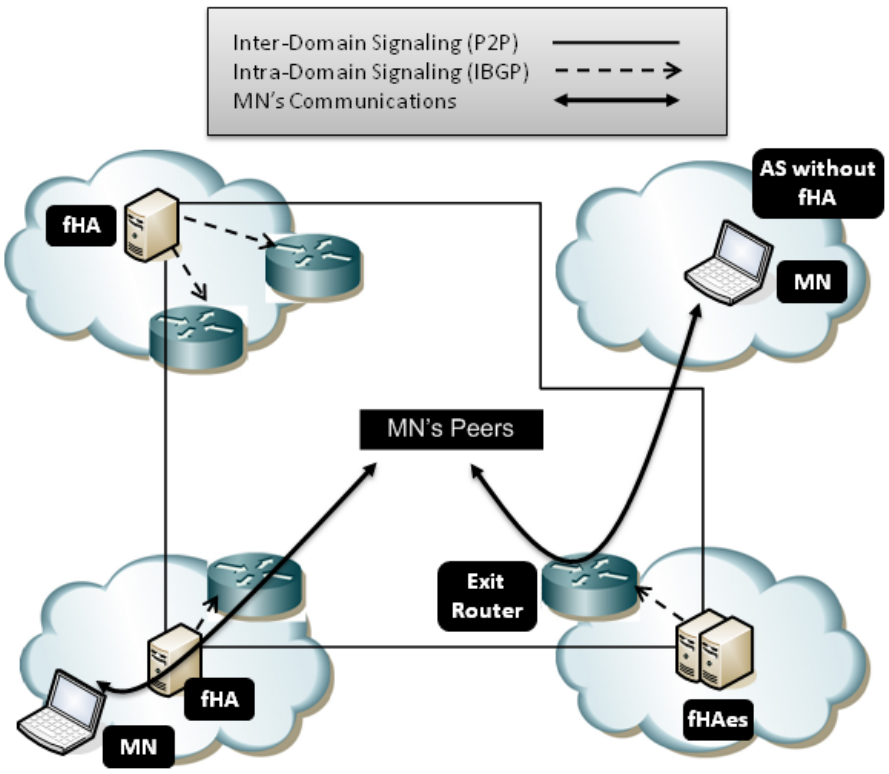


Figure 8.2: *Overview of the fP2P-HN architecture*

a simple metric able to capture the performance of a path.

Our proposal requires deploying several HAs throughout the Internet and has four differentiated phases. The HAs organize themselves in a Distributed Hash Table (DHT) overlay which stores the information regarding their IP addresses and their topological position (HA's AS number). This DHT is formed during the *P2P Setup phase*. The MNs are always bound to a HA belonging to this DHT. Thus, when the MN detects that the RTT to its current HA is unacceptable it triggers the *fHA Discovery phase* and queries the DHT for a close-by HA. Once the MN has the IP address of this close-by HA it sends a registration message (Binding Update) and obtains a new Home Address (HoA). We refer to this as the *fHA Registration phase*. The MN keeps using this HoA while the RTT remains below a given threshold.

All the HAs deployed in the fP2P-HN architecture are in fact flexible HAs. This means that they belong to the IBGP domain of their ASes. When their assigned MNs are attached directly to their AS they act as a regular HA. However, when the MNs are outside their AS, they announce the location of the MNs (Care-of Address) through IBGP to the AS Border Routers (BRs). This announcement is just a new route: To reach the MN (Home Address) packets must be addressed to its topological position (Care-of Address). This way, packets addressed from/to the MN are directly processed by the BR and thus, the load at the HA (and

also the intra-domain traffic) is considerably reduced. This is the last phase of the proposal known as *Data Packet Forwarding*.

Next we detail each one of the fourth phases.

8.4.2 P2P Setup Phase

This subsection details how the DHT is created. The DHT is used to store the location of the fHAs (AS number) and their IP addresses. This information is used by MNs to locate a close-by fHA to its topological position.

fHAs organize themselves forming a DHT. The fP2P-HN is fully flexible and can be deployed using any of the proposed DHT schemes [LCP+05]. In this work we consider Chord [SMLN+03] as DHT scheme, thus, the overlay's structure is a ring.

In the fP2P-HN, the search key is the *AS-key* that is computed as *hash(AS number)*. When a new fHA joins the fP2P-HN it chooses an identifier (Peer-ID). In our case this is the *hash(fHA's IP Address)*. The fHA's position in the ring is determined by its *Peer-ID*: the fHA is placed between the two overlay nodes with the immediately higher and lower *Peer-ID* to its own id.

Moreover, each fHA must register its AS number within the fP2P-HN. The fHA obtains the AS-key by computing the *hash(AS number)*. Then, it looks for the overlay node with the immediately higher *Peer-ID* to the *AS-key*, named *successor*, and sends to this node the *AS-key*, its IP address and its AS number. Moreover, the fHA sends some security information (See Section 8.4.7 for more details). The *successor* stores an entry with all this information, thus becoming the *Responsible Node* of this AS-key.

For a more detailed description of specific Chord functionality we refer the reader to Chapter 2.

8.4.3 fHA Discovery Phase (Inter-Domain)

This subsection details how an MN can use the fP2P-HN to discover a close-by fHA. The procedure is also depicted by Figure 8.3. A MN connected to a given fHA, fHA_1, eventually detects (after a handover) that the RTT to fHA_1 is above a given threshold. Then, it triggers the procedure to discover a close-by HA. The MN sends to its *Original fHA* (the one serving the MN's Home Network) a special BU soliciting the IP address of a close-by fHA. At this point, the *Original fHA* discovers (using BGP) the AS number associated to the MN's CoA. Afterwards, it obtains the *AS-key* by computing the *hash(AS number)*.

The search method within the fP2P-HN is as follows. The *Original fHA* sends a query with the AS-key. The search query is routed in the overlay towards the AS-key's *Responsible Node*. This fHA (e.g. fHA_2) is responsible of storing the information regarding the AS-key. Thus, it stores the IP addresses of all the fHAs located in the same AS as the MN. Then, fHA_2 sends these IP addresses to the *Original fHA* which in turn forwards them to the MN. Finally, the MN selects one of them and sends a special BU message to the new

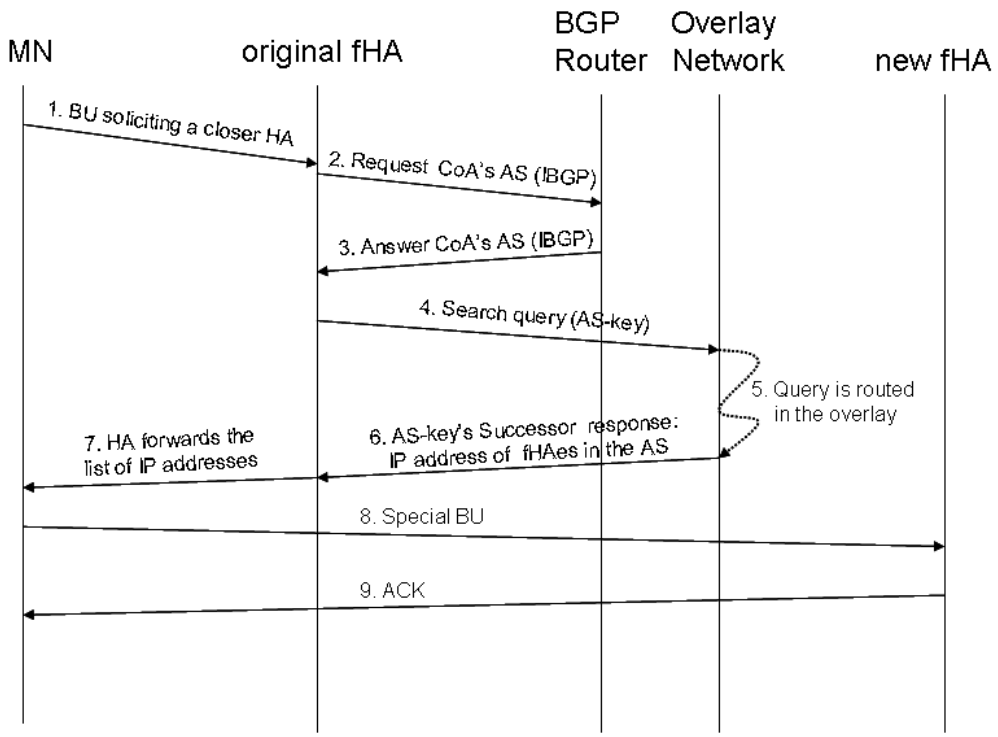


Figure 8.3: *fHA Discovery Phase in the fP2P-HN architecture*

fHA in order to obtain a new HoA. Several criteria can be used in order to select the fHA among those available on the same AS (load balancing, delay, etc). In this case, we select the fHA offering the lowest RTT that guarantees the security of the communications.

Although the fHAs are expected to be very stable entities, the fP2P-HN utilizes the mechanisms described in Chapter 5 to make the solution dynamic, adaptive and robust.

8.4.4 fHA Registration Phase (Intra-Domain)

This subsection details the registration phase of a MN into a new fHA. At the Intra-Domain level, each MN selects a given fHA through the above-mentioned mechanism. Our fHA has the same functionalities as a regular HA but it uses IBGP to signal the location of the MNs to reduce the load. The fHA acts just as a regular HA when the MN is directly attached to its network. However, when the MN is not directly attached to its AS, the fHA has to announce the new location of the MN (CoA) to the AS BRs. To distribute this type of information we use the Interior Border Gateway Protocol (IBGP). In the fP2P-HN, the fHAs and the BRs create an IBGP domain. This IBGP domain may be an already existing one or a separate one. The routes announced through this IBGP domain always have the longest prefix (/32) and never affect regular BGP routes. It should be noted that the routes announced by the fHAs are in any case distributed outside the AS. Finally, the entities participating in the IBGP domain have pre-configured keys to provide confidentiality, integrity

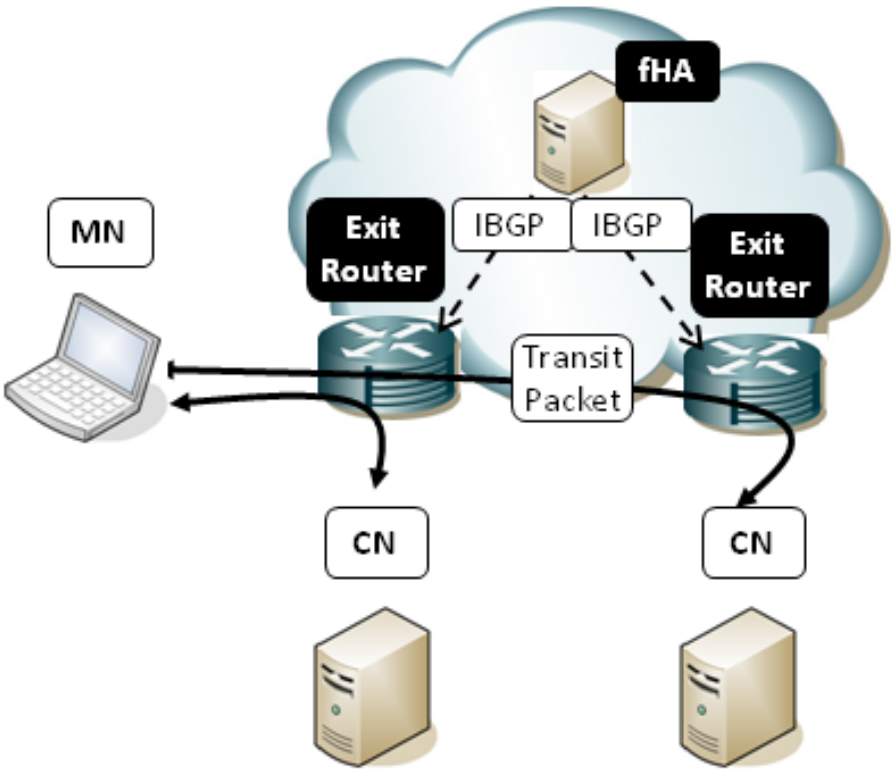


Figure 8.4: *Data Packet Forwarding*

and authentication for the communications.

For each received registration message (Binding Update) from outside the AS, the fHAs send an IBGP UPDATE message to the BRs. We introduce new options in the IBGP UPDATE message. The UPDATE message sent to the BRs includes the following information: $\langle$*Home Address, Care-of Address, Lifetime*$\rangle$. Upon reception of this message, the BRs setup a tunnel endpoint with the MN. The tunnel source address is the one of the BR's address while the destination address is the Care-of Address. In addition, each BR adds the following route to its routing table: $HomeAddress \backslash 32 \rightarrow Tunnel$. The tunnel and the route are automatically deleted after "Lifetime" seconds. Finally the fHA replies to the MN informing that the registration was successful and with the list of addresses of the BRs; this way the MN can address its tunnelled packets towards the BRs (see the next section for details).

Once the MN is assigned to a new fHA or returns home it sends a de-registration message to the previous fHA. Upon reception, the fHA sends an IBGP WITHDRAWAL message to the BRs to immediately remove all the routes and tunnels related to the MN's Home Address.

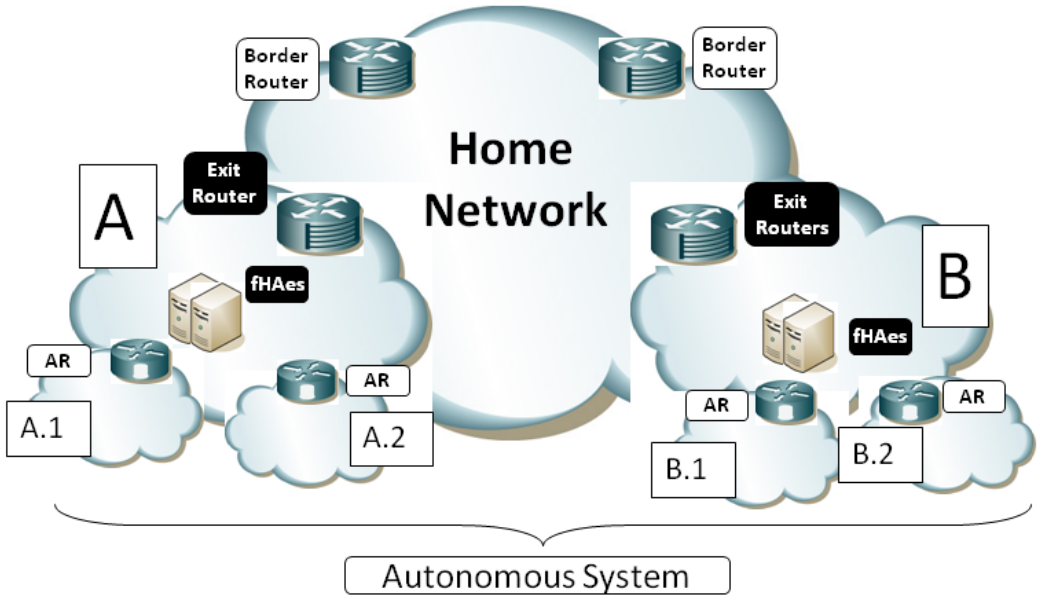


Figure 8.5: *Example of location of the fHAs*

8.4.5 Data Packet Forwarding Phase (Intra-Domain)

This subsection details how an MN's data packets are forwarded. If the MN is connected to the fHA's AS, then packets are forwarded just as in Mobile IP or NEMO. On the other hand, if the MN is attached to a foreign AS, then the MN should forward the packets through its HA. However, since the HA is a fHA, the MN encapsulate its data packets towards the BRs (figure 8.4). Since the fHA has previously configured (using IBGP) a new tunnel ($HomeAddress\backslash 32 \rightarrow Tunnel$) in the BRs, packets sent by the MNs are automatically de-capsulated and forwarded towards the packet's destination address (the MN's peer address). If the exit point of the MN's peer address is another BR, then the packet traverses the network as a transit packet.

Regarding the packets addressed towards the MN's HoA they reach the fHA's AS. The BRs have learned the location (CoA) of the MN through IBGP and automatically encapsulate and forward the packets directly towards the MN.

8.4.6 Flexible Home Agent Location

In the previous sections we have assumed -for clarity- that each fHA belongs to an IBGP domain with the AS BRs. However, our solution is flexible and allows that multiple sets of fHAs can be deployed in different networks of the AS. Then, each set of fHAs belongs to an IBGP domain with its network's *Exit Routers*. Figure 8.5 presents an example.

In this example, the AS has two different networks (A and B). Two different sets of fHAs

are deployed in network A and B. Thus, only routers labelled in black must belong to the IBGP domain with the fHAs of their network. The only requirement that these *Exit Routers* have to fulfil is being in the path of the packets addressed to the HoA delegated by the fHA.

An MN attached to this AS (A or B) is assigned to a given fHA; let's consider one located at the network A. Then, it receives a Home Address that belongs to the prefix of the network A. Thus, all the packets sent towards the MN will be received by the A's Exit Routers and forwarded directly to the MN. As noted previously, the MN encapsulates its data packets towards the A's Exit Routers that, in turn, de-capsulate and forward the packet towards the packet's destination address (the MN's peer).

8.4.7 Security Considerations

In Mobile IP and NEMO, the mobile clients and the Home Agents are under the same administrative domain. That is why they are equipped with pre-configured keys. These keys provide, among others, two essential security properties to the mobile communications, trustworthiness and confidentiality. This means that the MNs and the HA can trust each other since they are authenticated. Additionally, ciphering techniques can protect the communications.

However, the MNs of the fP2P-HN may connect to different fHAs that may or may not be under the same administrative domain. This section addresses the security at the fP2P-HN. Our goal is to achieve the same level of security as in Mobile IP and NEMO, that is: trustworthiness and confidentiality. In addition we also provide mechanisms to achieve a third security property, non-repudiation, but only when it is required.

It must be considered that security solutions are highly dependent on the application scenario. In this section we analyse security in two potential fP2P-HN scenarios: (i) the fP2P-HN is deployed by an unique organization and (ii) the fP2P-HN is formed by fHAs belonging to different organizations, typically Internet Service Providers (ISPs). In both scenarios, we address the security of the two types of communications present in the proposed solution: fHA-fHA and fHA-MN communications.

8.4.7.1 Scenario I: fP2P-HN deployed by an unique organization

In the first scenario, all the fHAs are deployed by the same organization. Several approaches can be used in order to provide fHA-fHA trustworthiness. For instance, all the fHAs own a X.509 certificate [CSF+08] provided by the organization that authorizes them to use the fP2P-HN services. This certificate provides trustworthiness, because any fHA can require another fHA's certificate in order to validate the latter as a legitimate entity. After being trusted, the fHAs involved in a communication can negotiate a shared key to provide confidentiality. This can be done by negotiating a session key based on a Public/Private keys pair generated by each fHA (A public key could be also included along with the certificate provided by the organization). Finally, non-repudiation is obtained if each fHA is required to sign every data packet with its private key.

For fHA-MN communication, MNs are granted with a credential from the organization in charge of the fP2P-HN. This credential allows unique identification of an MN in the system and could be provided in different ways: hardware device, SIM card, a user/password pair, a certificate, etc. Thus, in order to achieve trustworthiness, the MN obtains the fHA's certificate and the fHA requests the credential from the MN. Again, confidentiality is obtained by negotiating a session key between the MN and the fHA. Finally, if non-repudiation is required, it is achieved if fHAs sign the data messages using their private keys and MNs include their credentials within the messages.

8.4.7.2 Scenario II: fP2P-HN deployed by several organizations

This second scenario requires more complex security mechanisms because many different organizations are involved in the fP2P-HN deployment. Again, the most important requirements for the proposed solution are trustworthiness and confidentiality, but also non-repudiation is analysed.

We propose using a trusted third party (TTP) in order to achieve these goals. This TTP is trusted by all the organizations participating within the fP2P-HN and thus, by all the fHAs belonging to these organizations.

In this scenario, the organizations that offer mobility services are typically the ISPs. In addition, an ISP is represented by one (or more) ASes within the Internet architecture. Thus, we assume that a single ISP manages all the fHAs belonging to an AS.

In this architecture, each ISP participating in the fP2P-HN is granted with a X.509 certificate (per each managed AS) that is obtained from the TTP. This certificate contains, among other elements: the AS Number, the AS public key (AS_pu_key) and the valid period. It must be taken into account that each ISP has one AS private key (AS_pr_key) paired with each AS_pu_key. Then, all the fHAs deployed in a given AS use that certificate within the fP2P-HN. Only fHAs belonging to an ISP participating in the fP2P-HN are provided with such certificate. Therefore, based on this approach, we are able to provide the required security properties in the fHA-fHA communications.

Trustworthiness is achieved because only fHAs owning such a certificate (provided by the TTP) are trusted by the rest of fHAs within the fP2P-HN. Therefore, at any time a given fHA, fHA_1, could request from another fHA, fHA_2, its certificate to check whether fHA_2 is an authorized entity or not.

After both fHAs trust each other, they negotiate a shared key in order to provide confidentiality to the fHA-fHA communication. Several approaches could be applied at this point. For instance, the fHA_1 can provide a $nonce_1$[2] encrypted with the AS_pu_key$_2$ to the fHA_2, and similarly fHA_2. Therefore, both peers create a shared key using the nonces as input parameters to a given function. For instance, *Shared Key = f(*$nonce_1$*,*$nonce_2$*)* = $nonce_1$ *XOR* $nonce_2$.

In order to secure the fHA-MN communications, we propose a similar approach to that

[2] A nonce is a long random number.

used in GSM [gsm93b] [gsm93a] [gsm00] that validates users owning a SIM card using a credential. In GSM, when an user is attached to a foreign operator (roaming), it has to present its credentials to the new operator. Then, the new operator contacts the home operator and uses the received credentials to validate the user.

Following this approach, in the fP2P-HN the home AS (an ISP with the certificate provided by the TTP) provides credentials to its MN clients. This credential could be: a certificate, an unique ID like in GSM networks, etc. Therefore, once an MN selects a new fHA from a different ISP, it presents its credential and its home AS number to the new fHA. In turn, the new fHA validates the MN by sending to one of the fHA in the MN's home AS the credential. Then, based on the received credential, the fHA in the home AS checks if the credential's owner is an authorized user and returns the validation result to the new fHA. If the validation is successful the new fHA can trust the MN.

Finally, each MN has a permanent trusted connection with its *Original fHA*. Thus, the MN also trusts the new fHA because it has been authenticated by its *Original fHA*. This means that the new fHA is trusted by the *Original fHA* and also by the MN. Therefore trustworthiness is achieved in both directions. After that, a shared key could be negotiated between the fHA and the MN in order to provide confidentiality to the communications. Non-repudiation is achieved (if required) by applying the same mechanism introduced in the previous scenario.

8.4.8 Final Remarks

In this subsection we discuss the final considerations of the fP2P-HN. First, changing the MN's HoA may break the existing connections. In order to solve this issue we propose that these connections are forwarded through the previous fHA while new connections are forwarded through the new fHA. An MN changes its HoA only when it is outside of its currently assigned fHA's AS and the RTT is above a given threshold. ASes usually provide connectivity to very large geographical areas, thus, this will occur rarely. In addition, 98% of the connections last less than 15 minutes [BC02], this means that very few connections may be affected. Regarding the inbound connections, the MN may still use its original HoA (the one from its Home Network) or rely in dynamic reconfiguration protocols such as Dynamic DNS [Wel00] to make himself reachable in the new acquired HoA.

Second, the regular Mobile IP or NEMO handovers (i.e. changing the access router) are not affected by the fP2P-HN. That is, the procedural operations of the regular handovers are exactly as defined in the Mobile IP and NEMO standards. Therefore the latency of these handovers is the same in our approach as in Mobile IP or NEMO. Furthermore, the fP2P-HN adds a second handover type that occurs when the MN changes its HA and its HoA. Then the latency associated to these handovers is higher than in the regular one because it includes the search process in the DHT. However, since the existing connections are being forwarded through the previous HA, this extra handover latency does not affect the communications. Thus, we can conclude that although our solution introduces a new type of handover that suffers from a higher latency, this does not impact the performance of the communications.

Finally the architecture requires minor modifications in the MNs and HAs. Obviously, the HAs must include an implementation of the fHA and the DHT mechanisms. Regarding the MNs, they must include a triggering mechanism to discover a close-by HA. As noted previously, this mechanism can use any metric, although in this work we rely on the RTT. In addition, the MNs must support multiples HoAs, this is already under standardization by the MEXT WG [MEX]. The signalling between the MNs and the fP2P-HN can be accommodated into the Mobile IP signalling by exploiting the *Extensions* field present in the Binding Update messages (see [Per02] for details). Finally, the rest of the entities participating in the solution (CNs and routers) do not need to be modified. Since Mobile IP has not been deployed yet, we believe that the deployment cost of Mobile IP enhanced with the fP2P-HN would not increase.

8.5 Evaluation

The fP2P-HN architecture introduces two major improvements on Mobile IP and NEMO that are: the reduction in the delay of the communications and the reduction in the load at the HAs. However, these improvements increase the signalling load in both, Intra (IBGP) and Inter-domain (P2P) levels. In order to evaluate the advantages (*reduction in the communication's delay* and *reduction in the load at the fHAs*) and the costs (*Inter-Domain Signalling* and *Intra-Domain Signalling*) we have implemented the fP2P-HN in a simulator.

8.5.1 Simulation Setup

In order to simulate the proposed solution we have used Internet-like topologies generated with the last version (3.0) of *Inet* [JJJ$^+$00] [INE]. We have chosen *Inet* as the topology generator because it has been designed based on the analysis of public NLANR (National Laboratory for Applied Network Research) data-traces [pma]. These traces, well known by the passive measurements research community, have been collected from a variety of links at different networks. This means that *Inet* does not produce synthetic topologies, but realistic topologies based on real data-traces. In addition, *Inet* fulfils the requirements since it is intended to model AS-level connectivity instead of router-level connectivity. Regarding the mobility model, we have used the Random Waypoint Mobility simulator [PLBV05]. This simulator implements the well-known Random Trip Model [JBRAS03] that was proposed as a generic mobility model. We refer the reader to [PLBV05] and [INE] for further details.

Node-level simulators such as NS-2 or OMNET do not scale when simulating a large number of ASes. On the other hand AS-level simulators such as C-BGP or simBGP are not intended to include end-host mobility. That is why we have developed an ad-hoc simulator. We have implemented our simulator using Perl [Per]. The topology is generated using the *Inet* topology generator and the Random Waypoint Mobility model has been implemented into the simulator. The AS topology is stored as a graph using CPAN's *Graph* library and, for each MN, and after each movement, the shortest path to its fHA is computed using the Floyd-Warshall algorithm [CSRL01].

Armed with a topology generator and a mobility model we have developed an ad-hoc simulator. Unless noted otherwise, we have simulated an average number of 100 mobile clients per fHA. The MNs are distributed randomly (uniformly) among the fHAs, this means that the fHAs do not necessarily serve the same number of MNs. Each MN is assigned to a given Home Network (uniformly); the location of this Home Network is assigned randomly. For each handover, the MN has a 10% of probability of remaining in the same AS and, after a handover it remains attached to the same access router during a random amount of time distributed as (Gaussian) $N(5, 1)$ seconds. When the MN remains in the same AS, it means that it is changing its access router (CoA). Obviously, these values produce highly mobile nodes compared to the movements in real environments, however we aim to evaluate our solution in a stressful scenario. Regarding the delays of the links, we consider that each link has a constant delay uniformly distributed as $U[10, 25]$ms. Finally each MN sends 1 unit of bandwidth per second towards its Home Agent (for Mobile IP) and 1 unit towards its flexible Home Agent (for fP2P-HN). Since we aim to compare the load of both proposals a CBR data stream suffices. The MN's threshold to trigger the fHA discovery procedure is set to 75ms (This is based on the ITU-T recommendation defining an end-to-end delay lower than 150 ms for voice communications [itu96]).

We run each simulation during 1000 seconds (simulation time) running fP2P-HN and Mobile IP/NEMO. We consider the following deployment scenarios $\{0.01, 0.1, 0.3, 0.6, 0.75, 0.9\}$. These numbers represent the probability of deploying a fHA in a given AS. In the case of Mobile IP/NEMO, we consider the same number of HAs and the same number of MNs. Finally, we repeat the simulation of each deployment scenario 50 times with a different topology of 3500 ASes. The different topologies are generated using *Inet* (different seeds). In total, we have run 300 simulations. With this setup we simulate a wide range of scenarios, and we obtain the needed statistical information to assure the accuracy of the results. This accuracy is represented by the 90% Confidence Intervals included in every table and figure[3]. In order to run this huge amount of simulations we have used a cluster of 70 machines (Intel Xeon, 16Gb RAM) that uses Sun's N1 Grid Engine [clu].

The graphics included in this section represent the Cumulative Distribution Function[4] (CDF) of the different evaluated aspects and also provides the Confidence Intervals of the calculated CDF.

8.5.2 Simulation Results

8.5.2.1 Reduction of the Communication Delay

Firstly, we focus on the analysis of the communication delay since this is the main issue of Mobile IP and NEMO. Figure 8.6 shows the delay of the communications in the path between the MN and its current HA, both for Mobile IP and for the fP2P-HN. The figure presents the CDF of the average delay suffered from the communications of each MN. The

[3] In some figures the Confidence Intervals are so narrow that cannot be appreciated.

[4] In case of figure 8.7 the Complementary CDF is represented instead of the CDF.

results show that, for a very low deployment (1%), the fP2P-HN slightly outperforms Mobile IP/NEMO. However, increasing the deployment up to 10% leads to a reduction of the delay around 30%. This confirms, that even in the case of low deployments, our solution clearly outperforms Mobile IP or NEMO. Moreover, if we analyse the cases of higher deployments, the fP2P-HN reduces the communication delay up to 6 times compared to Mobile IP or NEMO.

Table 8.1 summarizes the results of figure 8.6. It shows the mean MN-HA communication delay for both fP2P-HN and Mobile IP/NEMO.

Deployment	fP2P-HN (ms)	Mobile IP (ms)	Reduction of the delay (%)
0.01	140.86 ± 0.95	145.83 ± 0.29	3.41%
0.10	112.12 ± 0.31	145.83 ± 0.29	23.12%
0.3	69.63 ± 0.16	145.83 ± 0.29	52.25%
0.6	40.77 ± 0.07	145.83 ± 0.29	72.04%
0.75	31.22 ± 0.04	145.83 ± 0.29	78.59%
0.9	25.93 ± 0.03	145.83 ± 0.29	83.25%

Table 8.1: *Mean MN-HA communication delay*

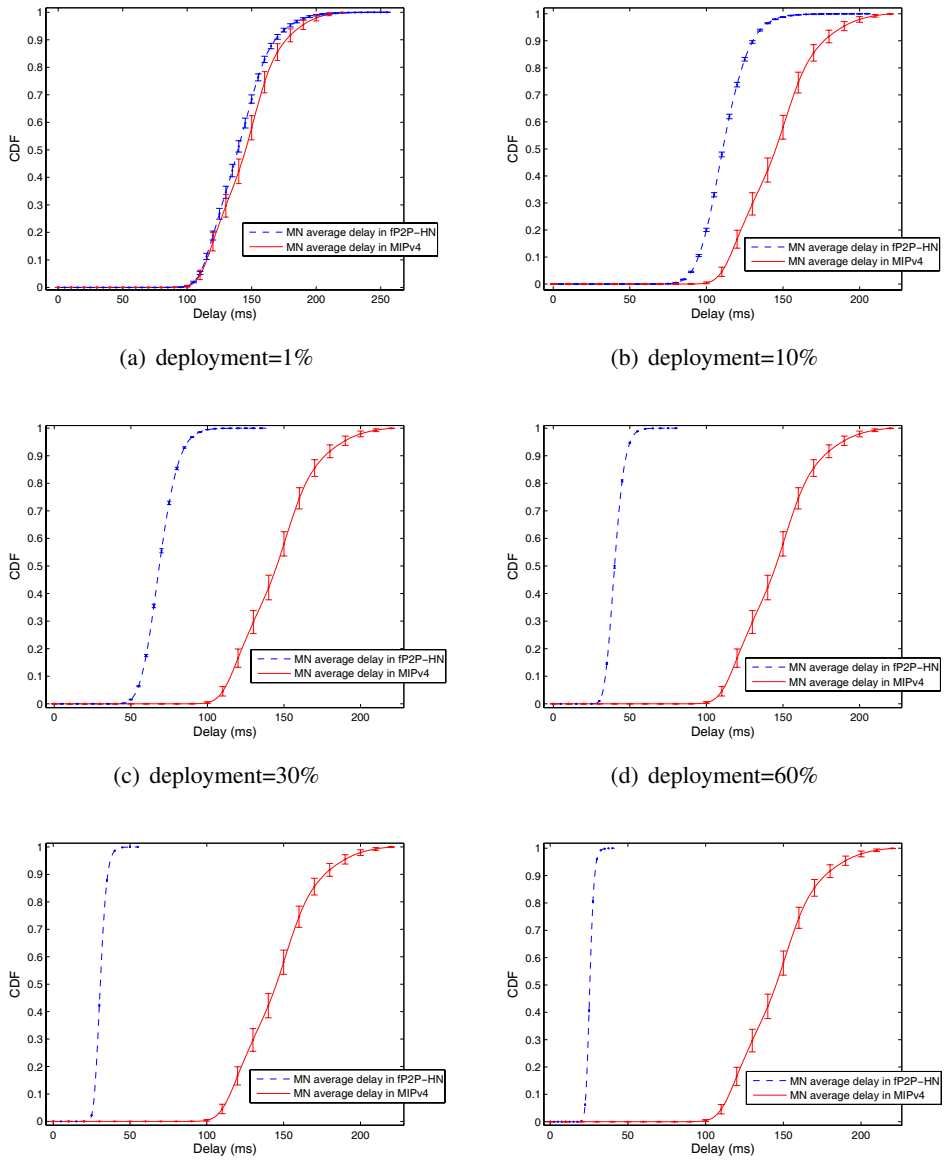
Thus, we can conclude that in terms of delay, fP2P-HN introduces a major improvement compared to the Mobile IP or NEMO solutions.

8.5.2.2 Reduction of the Load at the fHAs

In addition to the Route Optimization problem, the fP2P-HN addresses the reduction of the data traffic load at the HA as well. For this purpose we have introduced the concept of fHA. Figure 8.7 depicts the Complementary CDF (CCDF) of the percentage of saved traffic at the fHA compared to the regular Mobile IP's HA. The obtained results show that fP2P-HN introduces a major reduction of the load at the HA. The percentage of load reduction decreases along with the deployment. In the case of 1% of deployment we find that around half of the fHAs are free of data traffic load. This means that they delegate the forwarding task to the Exit Routers. Even considering large deployments (d=0.9), 80% of the fHAs experience a load reduction larger than 50% compared to Mobile IP/NEMO.

Table 8.2 shows the mean values. It must be noted that even in the worst case (d = 0.9) the mean load reduction with the fP2P-HN is 54.56%.

The reader may wonder why the percentage of saved traffic decreases as the deployment increases. This is because the fHAs delegates the forwarding of traffic from/to the MN when this is not directly attached to the fHA's AS. Whereas, if the MN is attached to its fHA's AS, then the fHA is responsible of forwarding the traffic from/to the MN. Hence if we consider a large deployment of fHAs, it is more likely that the MNs are located in the same AS as its fHA so that the fHA suffers from higher load. On the other hand, in case of low deployments, the probability that the MN finds a fHA in its current AS is lower. Then, the MN maintains the connection to the fHA located in a different AS which delegates the forwarding task to



(a) deployment=1%

(b) deployment=10%

(c) deployment=30%

(d) deployment=60%

(e) deployment=75%

(f) deployment=90%

Figure 8.6: *Average Communications Delay in the MN-HA path*

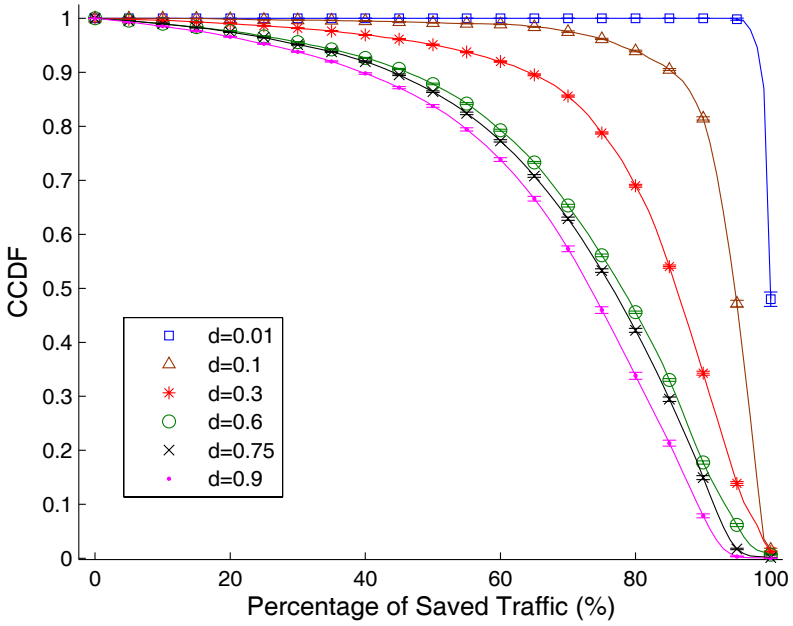


Figure 8.7: *Percentage of Saved Traffic per fHA regarding MIPv4*

the Border Routers. Thus the fHA's load is lower with low deployments.

In a nutshell, the higher the deployment, the higher the probability that an MN uses a fHA placed at its current AS; thus more data traffic is forwarded by the fHAs.

Deployment	Load Reduction (%)
0.01	99.31 ± 0.02
0.10	92.72 ± 0.03
0.3	78.94 ± 0.06
0.6	64.81 ± 0.04
0.75	59.35 ± 0.02
0.9	54.56 ± 0.72

Table 8.2: *Mean load reduction at the fHA compared to Mobile IP*

8.5.2.3 Inter-Domain Signalling

As explained above, existing solutions addressing the problem of Route Optimization for Mobile IP and NEMO may suffer from scalability problems. However the fP2P-HN uses a DHT (an scalable P2P technology) in order to signal the location of the HAs. In this section we evaluate the number of Inter-domain (P2P) signalling messages required to run the fP2P-HN.

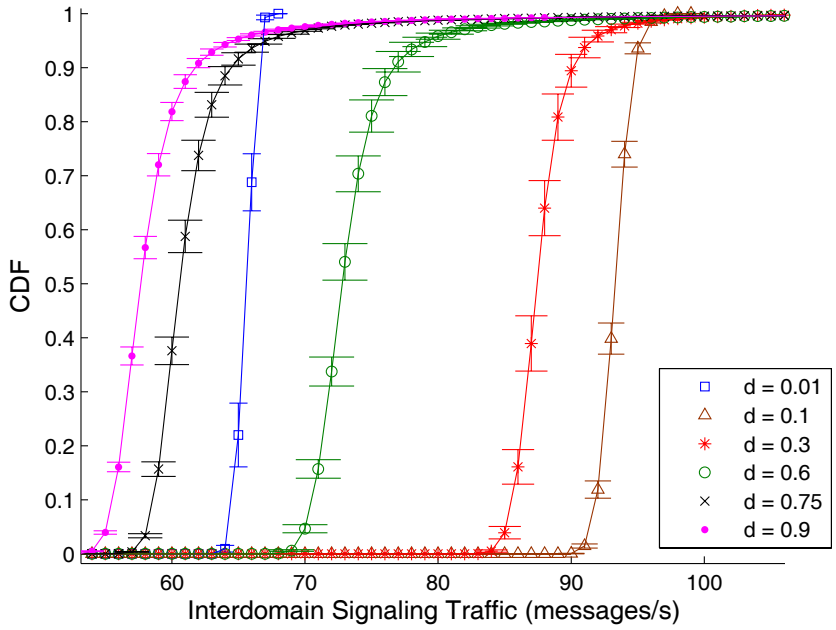


Figure 8.8: *fP2P-HN Inter-Domain signalling traffic*

Figure 8.8 shows the inter-domain (P2P) signalling generated by the fP2P-HN to signal the location of the different fHAs. This figure depicts the CDF of the number of inter-domain signalling messages per second (sent + received) that a fHA has to support in the fP2P-HN. We can observe that the signalling overhead introduced by the fP2P-HN remains between 50 and 100 messages/s for all the analysed deployments. Therefore, the fP2P-HN requires a low number of Inter-domain signalling messages. Moreover it must be considered that these messages are usually short messages; thus the bandwidth consumption is negligible. For instance if we consider the worst case of the figure (50 sent + 50 received messages per second) and we assume that each message has 50 bytes (a Mobile IPv4's Binding Update message has 44 bytes, see [Per02]); then the amount of signalling traffic that a fHA has to support in the fP2P-HN is 20 kbps (both uplink and downlink).

Table 8.3 presents the mean number of total messages/s supported by the fHA.

Deployment	Number of fHAs	Mean Number of (sent + received) messages/s
0.01	35	66.77 ± 0.14
0.10	350	94.46 ± 0.16
0.3	1050	89.23 ± 0.44
0.6	2100	75.21 ± 0.60
0.75	2625	63.32 ± 0.50
0.9	3100	67.63 ± 8.99

Table 8.3: *Mean Number of interdomain signalling messages/s per fHA*

Again it is worth analysing the signalling overload as function of the deployment. The reader can observe that the overhead increases as the deployment goes from 1% to 10%, and from this point it decreases along with the deployment increment. There are two parameters affecting the inter-domain signalling: the number of fHAs forming the fP2P-HN and the number of *special BUs* soliciting a new fHA (fHA discovery procedure). The number of fHAs has an influence since the fHA discovery procedure takes place at the overlay level and the query is routed by several fHAs within the fP2P-HN. The number of fHAs routing each query is bounded by $O(log_2(N))$ [LCP+05] (where N is the number of fHAs forming the fP2P-HN). Thus as deployment grows (larger N), more fHAs are involved in the routing of each query. On the other hand the number of *special BUs* gets reduced as the deployment increases. With large deployments is expected that MNs are always connected to close-by fHAs and that the fHA discovery process is rarely unsuccessful. Therefore, both parameters compensate each other. Thus when the deployment increases from 1% to 10%, the increment of the number of fHAs outweighs the reduction of the number of *special BUs* and the signalling load grows. For larger deployments the situation is reversed resulting in a signalling load reduction.

In order to further study this behaviour let's consider table 8.4. This table details the probability of triggering the fHA discovery procedure for each deployment scenario (the values have been collected from the simulations). As the table shows, when the deployment is low, the MNs initiate the fHA discovery procedure more often. This is because MNs detect that the RTT is above a given threshold, ask for a closer fHA, but, since deployment is low, do not find one. Hence the probability of triggering the fHA discovery procedure decreases as the deployment increases.

Deployment	Probability
0.01	0.73
0.10	0.64
0.3	0.47
0.6	0.35
0.75	0.29
0.9	0.27

Table 8.4: *Probability of triggering the fHA discovery procedure*

We can conclude that the fP2P-HN is scalable. Considering a highly mobile simulation scenario and 100 MNs per fHA, the signalling traffic in the worst case is around 20kbps. On the other hand, table 8.3 shows that the number of signalling messages is irrespective of the number of deployed fHAs. In fact independently of the deployment, the overhead is of similar magnitude (hundreds). Hence, the inter-domain cost of the proposed solution is $O(1)$.

8.5.2.4 Intra-Domain Signalling

Finally we analyse the Intra-Domain signalling. This signalling includes the IBGP (UPDATE and WITHDRAWN) messages sent to the *Exit Routers* and the BGP queries sent to

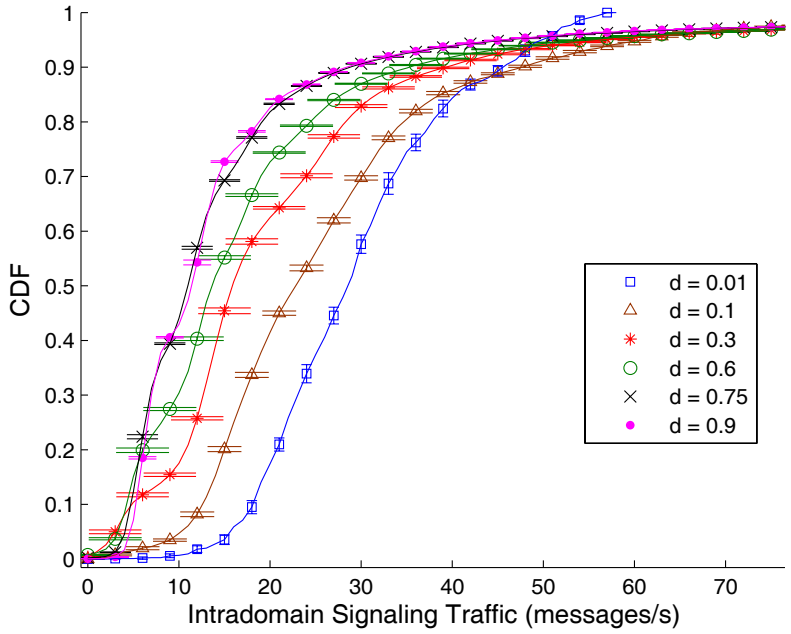


Figure 8.9: *fP2P-HN Intra-domain signalling traffic*

discover the MN's AS (see steps 2 and 3 in figure 8.3). This overload must be supported within each AS. Figure 8.9 shows the CDF of the amount of signalling per AS (per second), considering the different deployment scenarios. As the figure shows the number of signalling messages is bounded between 0 and 70 (sent + received) messages/s. Again, considering a message size of 50 bytes, the download/upload rate is less than 15 kbps. Additionally it has to be taken into account that this number is the total amount of signalling traffic supported inside each AS. Since the fP2P-HN allows deploying multiple fHAs within an AS (Sec. 8.4.6) each fHAs should only process a part of this signalling overload.

Table 8.5 shows the obtained mean values. The Intra-Domain signalling decreases as the deployment increases. This is an expected result, since when a MN is directly attached to a fHA in the same AS no IBGP signalling is produced.

Deployment	Average Number of (sent + received) messages (messages/s)
0.01	49.60 ± 0.03
0.10	45.96 ± 0.05
0.3	39.00 ± 0.09
0.6	32.57 ± 0.11
0.75	30.21 ± 0.12
0.9	29.24 ± 1.00

Table 8.5: *Mean Number of intradomain signalling messages/s per AS*

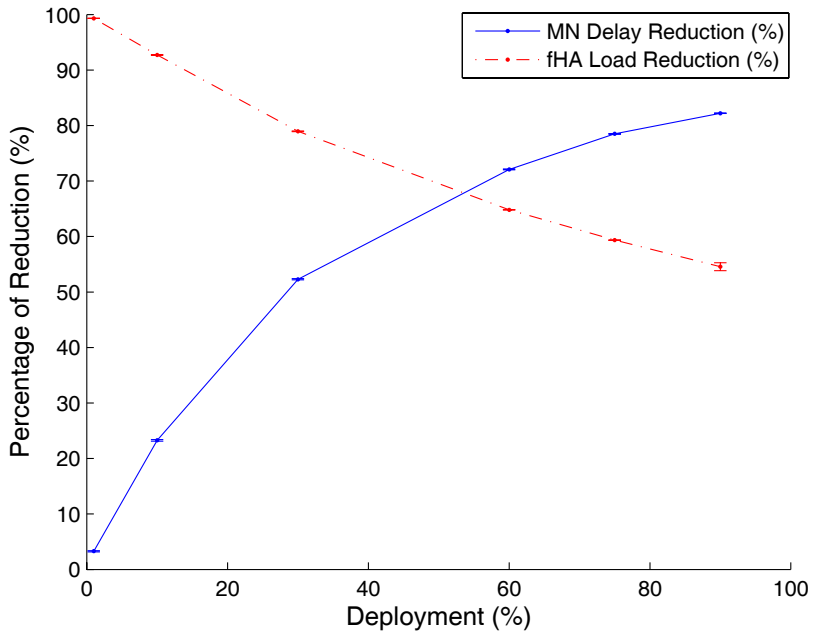


Figure 8.10: *Reduction of the communication's delay and fHA's load*

8.5.2.5 Summary of the obtained results

This section has evaluated the advantages and costs introduced by the fP2P-HN in front of the standard Mobile IP/NEMO protocols. The conclusion is that the fP2P-HN solves the main drawbacks of Mobile IP/NEMO (communication's delay and HA overload) with a low cost, some dozens of kbps in terms of extra signalling traffic. The obtained improvement depends on the deployment of the fP2P-HN. Figure 8.10 summarizes in a single graphic the improvements (load reduction and communication delay reduction) introduced by the fP2P-HN as function of the deployment. This figure allows us to determine the required deployment in order to achieve a given performance. For instance if we aim to reduce both the communication delay and the load at the HA over 60% then we should have a deployment between 45% and 65%. Finally large deployments improve the communication's delays while low deployments improve the reduction of the load at the fHAs.

8.6 Related Work

Incorporating route optimization to Mobile IP and NEMO clients is a key issue when considering the deployment of a truly mobile Internet. That is why this topic has attracted the attention of the research community and many solutions have been proposed.

The research community firstly focused on solving this problem specifically for Mobile

IPv4 [WCL+02] and NEMO clients [NTWZ07] [CBB+06] [NH04]. The main idea behind these proposals is to deploy a new entity at the correspondent network that helps the MN to communicate directly with the CN. Usually this new entity authenticates the location (CoA) and the identity (HoA) of the MN. In addition this device acts as a tunnel endpoint; this way the MN can send the packets tunnelled directly to the correspondent network. The main drawback of all these proposals is that they require deploying a new entity on each correspondent network. In the current Internet status this would imply deploying a new entity on each network or at least, on each AS (currently there are roughly 44.000 ASes on the Internet). We believe that the deployment cost of these solutions is too high.

R. Wakikawa presented recently a different approach [WOM05] used by other researchers [YHC06] [boe] [BGMBA06]. The key idea behind these proposals (as well as the fP2P-HN) is deploying multiple HAs in different Autonomous Systems (ASes). Then, an MN may pick the best HA according to its topological position thus, reducing the delay of the paths towards its peers. Some authors propose to use BGP to signal the location of the HAs [WOM05, boe, BGMBA06] whereas others use Anycast routing [YHC06]. These solutions may lead to scalability problems. On the one hand, using BGP protocol may produce an overhead in the already oversized global routing table [Hus01]. On the other hand, anycast's defiance of hierarchical aggregation makes the service hard to scale [KW00]. In addition, these solutions force the MNs to send the data packets through the HAs, increasing the load on these devices that may become the bottleneck of the whole system [NTWZ07]. In order to mitigate the aforementioned issues, the fP2P-HN uses a DHT (that is a fully scalable technology) to signal the HAs location and we benefit from the fHA that reduces significantly the load at the HAs.

8.7 Conclusions

The Mobile IP and NEMO protocols provide mobility for the Internet. Both protocols force the mobile nodes to send their data packets through a special entity (Home Agent) when communicating with their peers. This Home Agent is located at the mobile node's Home Network and forces the packet to follow a sub-optimal route. This reduces considerably the communications' performance, increases the delay and the infrastructure load. The research community has focused on solving this issue deploying several Home Agents throughout the Internet. Then a mobile node may pick a close-by one to its topological position in order to reduce the delay. Different authors use different technologies to signal the location of these Home Agents: eBGP, Anycast or a static list. Although these approaches reduce the delay they are not Internet-scalable. Additionally the Home Agents still have to forward all the mobile node's data packets and may become the bottleneck for the whole system.

In this chapter we have presented the fP2P-HN architecture, an adapted version of the solution presented in Chapter 5 addressing the problem of close-by HA discovery: (i) our architecture (as the aforementioned solutions) also deploys several Home Agents in order to reduce the delay; (ii) it uses a DHT overlay to signal the location of these Home Agents in an scalable way; (iii) the Home Agents of the architecture are in fact flexible Home Agents.

These agents signal the location of the mobile nodes within a network using the IBGP. By doing so the network's exit routers are aware of the location of the mobile nodes an can forward the packets by themselves, thus, the load at the flexible Home Agent is significantly reduced.

We have implemented the fP2P-HN in a simulator and we have evaluated the benefits and the costs of the architecture. The benefits are two: reduction of the delay and of the load at the Home Agents. The costs are the extra Inter and Intra domain signalling. We have put special attention on evaluating the Inter-domain overhead since this cost must be scalable. In order to provide significant results we have simulated the architecture using large Internet-like topologies of 3500 ASes and a mean number of 100 mobile nodes per Home Agent. Additionally each simulation has been repeated 50 times, using a different Internet-like topology, this way we can provide confidence intervals of the results. We tested different scenarios of deployment of the architecture, from 0.01 flexible Home Agents per Autonomous System to 0.9.

The main conclusions that can be extracted from the results are:

- The fP2P-HN effectively reduces the delay of the mobile nodes compared to Mobile IP/NEMO. Even with low deployments (0.1) the reduction is 23%. As the deployment grows so does the reduction that can be up to 83% (with a deployment factor of 0.9).
- Our architecture reduces the traffic processed by each flexible Home Agent compared to that of Mobile IP/NEMO. As expected, the reduction of the traffic decreases as the deployment increases. In the worst case (deployment factor = 0.9) the reduction of the average traffic processed by a flexible Home Agent is 54%. This reduction grows further to 99% with a deployment factor equal to 0.01.
- Our architecture is highly scalable since the amount of Inter-Domain signalling is always within the same order of magnitude (hundreds of messages per second) and it is irrespective of the number of flexible Home Agents deployed, thus, the cost is $O(1)$. Additionally the amount of Inter-Domain signalling traffic per flexible Home Agent is around 20kbps.
- The Intra-Domain signalling overhead of the fP2P-HN is very low, around 15kbps per AS. Since the architecture allows that multiple flexible Home Agents are deployed within an Autonomous System this overhead is shared among them.

Part III

Routing Fairness in Chord-like DHTs

Chapter 9

Routing Fairness in Chord

In this chapter we analyse the Routing Fairness in Chord. Furthermore, we propose a simple modification in the finger selection procedure of Chord that leads to a high improvement in the Routing Fairness.

We would like to highlight that this Chapter of the Thesis has been published in [CUnB09].

9.1 Overview

In the first part of this Thesis we have designed a simple and scalable solution that enables the dynamic and location aware server discovery. The servers participating in our solution suffer from an overhead due to their participation in a Chord DHT. Unfortunately, this overhead is intrinsic to the solution and we cannot avoid it. Then, it is important to distribute this overhead fairly among the servers. In our system as well as in other DHT-based systems (e.g. P2PSIP) the size of the stored objects is small, thus, routing dominates the cost of publishing and retrieving an object. In such systems, the issue of fairly balancing the routing load among all nodes becomes critical. In this second part of the Thesis we address this issue for Chord-based P2P systems. We first present an analytical model to evaluate the routing fairness of Chord based on the well accepted Jain's Fairness Index (FI). Our model shows that Chord performs poorly, with a FI around 0.6, mainly due to the different sizes of the zones between nodes. Following this observation, we propose a simple enhancement to the Chord finger selection algorithm with the goal of mitigating this effect. The key advantage of our proposal as compared to previous approaches is that it adds a negligible overhead to the basic Chord algorithm. The proposed approach is evaluated analytically showing a very substantial improvement over Chord, with a FI around 0.9. We conduct an extensive large-scale simulation study to evaluate our proposal and validate the analysis. The simulation study includes, among other aspects, churn conditions, heterogeneous nodes and Zipf-like object popularity. In all of the cases our solution outperforms Chord.

9.2 Introduction

Peer-to-Peer (P2P) systems have become one of the most popular applications in the Internet, mainly pushed by file sharing applications such as BitTorrent and Emule, in addition to emerging applications such as peer-to-peer SIP (P2PSIP) [p2p]. Indeed, nowadays P2P applications are responsible for most of the Internet traffic [Ipo07]. P2P systems are classified into unstructured P2P systems, such as Gnutella [KM02], and structured P2P systems or Distributed Hash Tables (DHTs), such as Chord [SMLN+03], Kademlia [MM02] or Pastry [RD01]. The main advantage of DHTs is that data placement and search procedures generate less traffic. Among the DHT-based approaches, Chord is one of the most popular systems[1] and it is the focus of the present study.

Although the issue of fairly balancing the load among the nodes of a DHT has been extensively studied in the literature [GLS+04, RLS+03, GS05, ZH04, ZH05, BCM03, KM05, KR04, LS05, SVF08], most of the efforts so far have been devoted to fairly distributing the load of stored data, while the issue of balancing the routing load has received much less attention. However, as it has been observed by [GS05], in DHT-based P2P systems where objects are small, routing dominates the bandwidth and latency costs of storing and finding an object. This is the case for the system proposed in the first part of this Thesis (the stored objects are list of IP addresses) and also for most of the existing P2P indexing systems where the objects fetched from the DHT are just lists with the IP addresses of the hosts that actually store the desired content (e.g KAD). P2PSIP is another clear example of a P2P system with small objects. As the load of such systems is dominated by routing, it becomes critical to fairly balance the routing load among nodes.

To the knowledge of the authors, the only work that has explicitly addressed the issue of routing fairness in DHTs is [SBKF07]. This proposal is specific to Pastry and its key idea is to dynamically remove from the routing tables those nodes that are responsible for the most popular objects. One disadvantage of this approach is that it incurs additional overhead and complexity due to the dynamic maintenance of the routing tables. Additionally, the proposals made to balance the load of stored data may be also applied to the routing load; however, these also incur additional overhead. In particular, [GLS+04, RLS+03, GS05, ZH04, ZH05] assign several logical nodes (called virtual servers) to physical nodes, which adds an overhead proportional to the number of virtual servers. Finally, other approaches [BCM03, KM05, KR04] aim at modifying the node identifiers to keep them equally spaced, which adds a high overhead to update ids when nodes join and leave the ring. For a more detailed description of the Related Work we refer the reader to Section 3.2.

In this part of the Thesis we address the issue of balancing the routing load of Chord. We first analyse the routing fairness of Chord. Then, based on this analysis, we propose a simple enhancement to the finger selection mechanism that improves very substantially Chord's fairness. The main advantage of this approach over existing proposals is that it adds a negligible overhead to the standard Chord algorithm. The key contributions of this chapter are:

[1] For instance, Chord has been recently adopted as mandatory by the IETF P2PSIP Working Group [p2p].

- We analyse the routing fairness of Chord according to the Jain's Fairness Index [JCH98]. Although the goal of our analysis is to evaluate fairness, the analysis also contains other findings that are valuable contributions by themselves, including the characterization of the size of the zone between nodes, the number of fingers and the number of routed messages.

- We propose an enhancement to the basic Chord finger selection mechanism that improves very substantially the fairness of the routing load by adding a negligible extra overhead. The fairness level of the proposed mechanism is studied analytically.

- We run large scale simulations, with up to 10^6 nodes, that validate our analysis as well as the performance improvement obtained with the proposed Chord enhancement.

The rest of this chapter is organized as follows. Section 9.3 presents an analysis of the routing fairness of Chord. Based on this analysis, we identify the main cause for Chord unfairness, which motivates the enhancement that we propose. This enhancement is described and analysed in Section 9.4. Section 9.5 then evaluates the performance of Chord and the proposed mechanism, both analytically and via simulation. Finally, Section 9.6 closes the chapter with some final remarks.

9.3 Analysis of the Routing Fairness in Chord

In this section we analyse the fairness of the routing in Chord[2]. The metric that we use to this aim is the Jain's Fairness Index (FI) [JCH98], which is a widely accepted metric to assess the level of fairness of a distribution. This metric gives a value between 0 and 1, where 0 is the unfairest distribution and 1 is the fairest one. Applying Jain's Fairness Index to the routing load of each node in the ring (measured in terms of the number of routed messages) results in the following expression:

$$FI = \frac{|\sum_{i=1}^{n} m_i|^2}{n \sum_{i=1}^{n} m_i^2} \tag{9.1}$$

where n is the number of nodes in the ring and m_i is the number of messages routed by node i.

Our analysis of the routing fairness proceeds along the following steps:

- We first analyse the distribution of the size of the zone between two consecutive nodes of the Chord ring (i.e. the zone of responsibility of a node).

- Then we analyse the number of *inbound fingers* pointing to a node as a function of the size of the zone of responsibility of that node and, from this, the distribution of *inbound fingers*.

[2]For a detailed description of Chord functionality we refer the reader to Section 3.2

- Next, we obtain the number of messages routed by a node as a function of its number of *inbound fingers*.
- Finally, we compute the FI of the routing load based on the distribution of the number of messages routed by one node.

Without loss of generality, our model assumes that node identifiers are randomly distributed in the continuous id-space [0,1]. This is an accurate approximation of the real scenario in which the node-ids are randomly distributed in the discrete id-space [0,2^k), where k is the length of the hash operation employed to generate identifiers (e.g. 128, 160 or 256). Note that with such large values of k, the assumption that the id-space is continuous instead of discrete yields very accurate results. The conversion of the space [0,2^k) to [0,1] only introduces a scalar factor whose only purpose is to simplify the notation.

Let us start with the analysis of the size of a node's zone of responsibility in Chord. Let X be the random variable that represents the size of the zone between two consecutive nodes. The probability of the node's zone (X) to be smaller than a given value x is equal to the probability of having at least one node inside X. Since nodes ids are uniformly distributed in the ring and the id-space size is equal to 1, the probability that a node falls within a zone of size x is precisely x. This yields the following cumulative distribution function for random variable X:

$$cdf\ (x) = P\,(X \leq x) = 1 - (1 - x)^n \tag{9.2}$$

The probability distribution function of the size of the node's zone of responsibility, $pdf(x)$, is simply computed as the derivative of the above cdf:

$$pdf(x) = n(1 - x)^{n-1} \tag{9.3}$$

We next model the probability $P(f|x)$ that a node has f *inbound fingers* given that it has a zone of size x. Our key approximation in the modelling of $P(f|x)$ is to assume that all the fingers in the ring are uniformly distributed in the id-space. From this assumption, the probability that a node whose zone size is x has f inbound fingers can be computed as the probability that f fingers fall into a zone of size x;

$$P(f|x) = \binom{F}{f} x^f\,(1 - x)^{F-f} \tag{9.4}$$

where F is the total number of fingers in the ring.

To compute the above expression, we need to obtain F, which we do as follows;

$$F = nf_{out} \tag{9.5}$$

where f_{out} is the average number of *outbound* fingers of Chord nodes.

To calculate f_{out}, we consider the following behaviour of Chord. A node with a given id looks for outbound fingers in id-space $[id, id + 2^k)$ as follows. It divides the id-space in k zones, namely $[id + 2^i, id + 2^{i+1})$, with $i = 0, 1, 2, ..., k - 1$. Then, it establishes an outbound finger to one of these zones if the following two conditions hold:

- There should be at least one node in the zone $[id + 2^i, id + 2^{i+1})$, as otherwise no node can be selected as a finger. This occurs with probability:

$$P_{node}(i) = 1 - \left(1 - \frac{2^i}{2^k}\right)^n \tag{9.6}$$

- There should be at least s successors in the zone $[id, id+2^i-1]$, i.e. between the node id and the given zone, as otherwise this will be a successor link and not an outbound finger. This occurs with probability:

$$P_{succ}(i) = 1 - \sum_{j=0}^{s-1} \binom{n}{j} \left(\frac{2^i - 1}{2^k}\right)^j \left(1 - \frac{2^i - 1}{2^k}\right)^{n-j} \tag{9.7}$$

Assuming that the above conditions are independent, we can compute the average number of *outbound* fingers of a node as the sum across all the zones of the probability that the node has an outbound finger pointing to a node in the zone:

$$f_{out} = \sum_{i=0}^{k-1} P_{node}(i) P_{succ}(i) \tag{9.8}$$

The above terminates the analysis of $P(f|x)$. The next step is to compute the distribution of the number of inbound fingers, $P(f)$. We do this by combining Eqs. (9.3) and (9.4)[3],

$$\begin{aligned} P(f) &= \int_0^1 P\left(f|x\right) pdf\left(x\right) dx \\ &= \int_0^1 \left(\binom{F}{f} x^f \left(1-x\right)^{F-f}\right) \left(n\left(1-x\right)^{n-1}\right) dx \\ &= \binom{F}{f} n \int_0^1 x^f \left(1-x\right)^{F+n-f-1} dx \\ &= \binom{F}{f} n \frac{f!\left(F+n-f-1\right)!}{\left(F+n\right)!} \end{aligned} \tag{9.9}$$

We next address the analysis of the average number of messages routed by a node (m) as a function of the number of inbound fingers of that node (f). The key approximation here is to assume that all successor links in the ring route the same number of messages and so do all fingers. With this approximation the number of messages routed by a node can be expressed as a function of the number of *inbound* fingers pointing to this node as follows[4];

$$m(f) = (s-1)M_s + (f+1)M_f \tag{9.10}$$

[3]To solve the integral we have used $\int_0^1 x^a(1-x)^b \delta x = \frac{a!b!}{(a+b+1)!}$.

[4]We account for the last successor as a finger instead of a successor because of the following. The number of messages that a node routes through a link depends on the size of the zone between this link and the next one. Since successors are close to each other, this size will be small for all successors but the last one. Indeed, the link next to the last successor is a finger, and the zone inbetween is larger than the zone between two successors. Following this, in our analysis we take $s-1$ for the number of successors and $f+1$ for the number of fingers.

where M_s is the average number of messages routed by each successor link and M_f the average number of messages routed by each finger.

In the above expression, M_s and M_f remain to be computed. We start with M_s. Let q be the total number of queries routed in the ring during a given observation interval, and H_s the average number of hops per query over successors links. Then, M_s can be computed as the total number of hops routed over successors links during the interval, qH_s, divided by the total number of successor links in the Chord ring, $n(s-1)$, which yields:

$$M_s = \frac{qH_s}{n(s-1)} \tag{9.11}$$

In order to compute H_s in the above expression we proceed as follows. When routed towards a certain destination, a query message may either reach the destination node directly via a finger, or it may first reach one of the $s-1$ nodes that has the final destination in its successors list. In the former case, there isn't any hop in successors links, while in the latter there is one. Assuming that any of these s nodes are reached with the same probability, the second event occurs with probability $(s-1)/s$, which yields:

$$H_s = \frac{s-1}{s} \tag{9.12}$$

To compute M_f we proceed similarly to Eq. (9.11): we divide the total number of hops routed over fingers by the total number of fingers in the ring, which gives

$$M_f = \frac{qH_f}{n(E(f)+1)} \tag{9.13}$$

where H_f is the average number of hops per query routed over fingers, and $E(f) = F/n$ is the average number of fingers per node.

To compute H_f we follow the result of [SMLN+03] which shows that the average number of hops in a Chord ring that does not use successor links to route queries is given by $\frac{1}{2}log_2(n)$, where n is the number of nodes of the ring. In order to apply this result in a ring that uses successor links, we assume that this expression holds when considering a subportion of the ring. In particular, we consider that the number of hops using fingers required to cover a zone with s successors is given by $log_2(s)/2$. Since, by using successor links we save the hops required to cover this zone, these do not have to be counted in H_f, which gives

$$H_f = \frac{log_2(n)}{2} - \frac{log_2(s)}{2} \tag{9.14}$$

The above terminates the analysis of $m(f)$; combining all the equations we obtain:

$$m(f) = \frac{q}{n}\left(H_s + (f+1)\frac{H_f}{E(f)+1}\right) \tag{9.15}$$

In summary, our analysis has shown that a node routes a number of messages $m(f)$ given by Eq. (9.15) with a probability $P(f)$ given by Eq. (9.9), for $f = 0, 1, 2, \ldots, F$.

With the above, we can proceed to compute the Fairness Index (FI) given by Eq. (9.1), which is the objective that we set at the beginning of this section. The numerator of Eq. (9.1) corresponds to the square of the total number of messages routed. Since the total number of queries is given by q, and each query takes $H = H_s + H_f$ hops in average, we have:

$$\left| \sum_{i=1}^{n} m_i \right|^2 = (qH)^2 \tag{9.16}$$

To compute the denominator of Eq. (9.1), we rearrange the sum by grouping all the nodes that have f fingers (and therefore route $m(f)$ messages according to our previous analysis) together:

$$n \sum_{i=1}^{n} m_i^2 = n \sum_{f=0}^{F} n_f m^2(f) \tag{9.17}$$

where n_f is the number of nodes that have f inbound fingers. Since the probability that a node has f fingers pointing to it is given by $P(f)$, we have:

$$P(f) = \frac{n_f}{n} \tag{9.18}$$

from which:

$$n \sum_{i=1}^{n} m_i^2 = n^2 \sum_{f=0}^{F} P(f) m^2(f) \tag{9.19}$$

Combining Eqs. (9.1), (9.16) and (9.19) we obtain:

$$FI = \frac{(qH)^2}{n^2 \sum_{f=0}^{F} P(f) \left(\frac{q}{n} \left(H_s + \frac{H_f}{E(f)+1} (f+1) \right) \right)^2} \tag{9.20}$$

from which we finally obtain the following expression:

$$FI = \frac{H^2}{H_s^2 + \sum_{f=0}^{F} P(f) \left(\left(H_f \frac{f+1}{E(f)+1} \right)^2 + 2 H_s H_f \frac{f+1}{E(f)+1} \right)} \tag{9.21}$$

which terminates the analysis routing fairness in Chord.

9.4 Enhanced Chord: Proposal and Analysis

Following the analysis presented above, it can be observed that the routing unfairness of Chord is caused by the different size of the zones between nodes. Indeed, since ids are randomly placed in the ring, it is well known that zones will typically have very different sizes [SMLN+03]. Then, the larger the size of the zone of responsibility of a node, the larger the probability that a finger falls within this zone and therefore the more inbound fingers this

node will have. Since the number of messages routed by a node grows with the number of fingers that point to the node, this yields a large degree of unfairness. In this section, we propose (and analyse) an enhancement of Chord that improves the routing fairness without changing the distribution of nodes and zones between them. We name our solution Enhanced Chord (e-Chord).

Since, according to the above, routing unfairness in Chord is caused by the unbalanced distribution of inbound fingers, the main goal of e-Chord is to modify the finger selection algorithm in order to achieve a more balanced distribution. With Chord's original finger selection algorithm, a node with a larger zone is more likely to be chosen as a finger ; in order to avoid this, the key idea of e-Chord is the following. When a node is selected to be a finger, instead of pointing the finger to this node, we choose with some probability one of the node's successors and point the finger to this successor. In particular, when choosing a finger, the node and all its successors are selected with exactly the same probability. In this way, the number of inbound fingers of a node does no longer keep a strong correlation with its zone size, since a node with a large zone will share incoming fingers with its successors.

In more detail, the finger selection algorithm of e-Chord works as follows. Like in basic Chord, when a node with an identifier id creates its fingers table, it starts by performing a lookup to know which are the nodes n_i with an identifier equal to or greater than $key_i = id + 2^{i-1}\ \forall i \in [1, k]$. Then, instead of selecting this node as the finger (as it would be done in basic Chord), the finger is picked up randomly from the set of $R = s + 1$ nodes formed by node n_i and its s successors.

Note that this mechanism does not require e-Chord nodes to keep any additional information than with standard Chord, since nodes only need to know their s successors. Moreover, the random process for finger selection of e-Chord can be performed at the remote node n_i in a transparent way without adding any extra overhead. In particular, when a node issues a query to set up a finger and this query reaches the destination node, instead of returning the IP address of n_i, it can simply return with some probability the address of one of its successors.

Another advantage of the proposed solution is that it does not require to modify any other Chord mechanism and in particular the Chord lookup algorithm is not modified at all. Additionally, e-Chord also keeps the $O(log(n))$ performance of the Chord lookup algorithm. Indeed, this performance is given by the fact that fingers are located in different 2^i zones of the Chord ring and does not depend on the specific nodes being used as fingers within these zones.

In summary, with a simple modification to the standard Chord finger selection mechanism, e-Chord improves the fairness of the routed messages without adding any control traffic overhead. We next analyse the resulting routing performance; the analysis follows the same lines as the analysis of the routing fairness of Chord in the previous section, although there are some significant differences:

- Instead of computing the distribution of the zone between one node and the next one, we now compute the distribution of the zone that contains a node and its s successors. We refer to this as a *R-node zone*, where $R = s + 1$ is the number of nodes contained

in this zone.

- The probability that a node is chosen as a finger is computed as $1/R$ times the probability that it falls into the node's R-zone.
- Once the number of inbound fingers that point to a node has been computed, the rest of the analysis proceeds like for the Chord case.

To analyse the distribution of an R-node zone we proceed as follows. Let its size be represented by the random variable X. Then, the probability that this variable is smaller than a give value x is equal to the probability that more than R nodes have their identifiers within this zone. Since the node ids are uniformly distributed along the $[0, 1]$ id-space, the probability that one given id falls in this area is x, which yields:

$$cdf\,(x) = Pr\,(X < x) = 1 - \sum_{r=0}^{R-1} \binom{n}{r} x^r\,(1-x)^{(n-r)} \tag{9.22}$$

The pdf of the R-node zone is computed as the derivative of the above:

$$pdf\,(x) = \sum_{r=0}^{R-1} \binom{n}{r} (nx - r)\,x^{(r-1)}\,(1-x)^{(n-r-1)} \tag{9.23}$$

For a finger to point to a given node whose R-node size is x, the following two things must happen: i) the finger key has to fall within the node's R-zone, which occurs with probability x, and ii) the node must be selected among the R possible candidates for this finger, which occurs with probability $1/R$. This yields a probability of x/R for a node to be chosen as a finger. Following this, we can compute the probability that a node has f *inbound* fingers as follows:

$$P(f|x) = \binom{F}{f} \left(\frac{x}{R}\right)^f \left(1 - \frac{x}{R}\right)^{(F-f)} \tag{9.24}$$

The next step is to compute the distribution of the number of inbound fingers in e-Chord, $P(f)$:

$$\begin{aligned} P(f) &= \int_0^1 P(f|x) pdf(x) dx \\ &= \sum_{r=0}^{R-1} \binom{n}{r} \binom{F}{f} \frac{1}{R^f} \int_0^1 (nx - r)\,x^{r+f-1} (1-x)^{n-r-1} \left(1 - \frac{x}{R}\right)^{F-f} dx \end{aligned} \tag{9.25}$$

By applying the following two equalities to the above,

$$(1-x)^z = \sum_{k=0}^{z} \binom{z}{k} 1^k\,(-x)^{z-k} \tag{9.26}$$

$$\left(1-\frac{x}{R}\right)^z = \sum_{k=0}^{z} \binom{z}{k} 1^k \left(\frac{-x}{R}\right)^{z-k} \tag{9.27}$$

we obtain:

$$\begin{aligned} P(f) = & \sum_{r=0}^{R-1} \sum_{k=0}^{n-r-1} \sum_{j=0}^{F-f} \binom{n}{r} \binom{F}{f} \binom{n-r-1}{k} \binom{F-f}{j} \frac{1}{R^{F-j}} (-1)^{n-r-1-k+F-f-j} \\ & \int_0^1 (nx-r)\, x^{F+n-k-j-2} \delta x) \\ = & \sum_{r=0}^{R-1} \sum_{k=0}^{n-r-1} \sum_{j=0}^{F-f} \binom{n}{r} \binom{F}{f} \binom{n-r-1}{k} \binom{F-f}{j} \frac{1}{R^{F-j}} (-1)^{n-r-1-k+F-f-j} \\ & \left(\frac{n}{n+F-k-j} - \frac{r}{n+F-k-j-1}\right) \end{aligned} \tag{9.28}$$

Once $P(f)$ has been obtained, the rest of the analysis to compute the Fairness Index of e-Chord routing follows the same Eqs. (9.10) – (9.21) of the basic Chord analysis. This terminates the analysis of the routing fairness in e-Chord.

9.5 Performance Evaluation

In this section we evaluate the performance of the basic Chord mechanism and the proposed enhancement in terms of routing load and fairness. Furthermore, we validate the presented analytical model by comparing it against simulations results. Simulations have been performed using our own simulator developed in Java[5]. For all simulations, 95% confidence intervals are given with errorbars (note that in many cases they are so small that they can barely be appreciated in the graphs). Unless otherwise stated, we take a number of nodes $n = 10^6$ and a number of successors $s = 16$ as default values. Note that these are typical settings in real Chord implementations [DLS+04] and real P2P applications [SENB07]. The number of queries in each simulation run is $q = 10^8$, and for each query message a pair of source and destination nodes is chosen randomly among all nodes (except for the experiment of Section 9.5.7 in which we consider a Zipf-like object popularity distribution).

9.5.1 Routing fairness

We first evaluate the routing fairness of Chord and e-Chord. For this purpose, we use the Jain's Fairness Index (FI) applied to the routing load as proposed in Eq. (9.1). Table 9.1 gives the FI obtained via analysis and simulation for different settings, in the ranges $n \in [10^3, 10^6]$ and $s \in [8, 32]$. We draw the following conclusions from these results:

[5]The source code of the simulator is available under a GPL license at `http://www.it.uc3m.es/rcuevas/infocom09/ChordSim.jar`.

n	s	chord simulation	chord analysis	e-chord simulation	e-chord analysis
10^3	16	0.6470	0.6726	0.9029	0.9109
10^4	16	0.6024	0.6166	0.8996	0.9126
10^5	16	0.5752	0.5882	0.9039	0.9163
10^6	16	0.5594	0.5710	0.9064	0.9198
10^6	8	0.5596	0.5638	0.8816	0.8886
10^6	24	0.5591	0.5746	0.9149	0.9309
10^6	32	0.5618	0.5772	0.9189	0.9360

Table 9.1: *Fairness Index of the messages routed by Chord/e-Chord nodes*

- Simulation follow analytical results fairly closely, with small errors of around 1% and 2% both for Chord and e-Chord. This validates the analyses presented in Sections 9.3 and 9.4.

- The proposed e-Chord scheme outperforms Chord very substantially. Indeed, the fairness index provided by Chord is around 0.6 for all n and s settings, while e-Chord provides a much greater fairness index, around 0.9. This validates the proposed approach as it provides a much better routing fairness than standard Chord.

9.5.2 Routing load distribution

The results given in the previous section allow to assess the overall level of fairness of the routing load but do not show how the routing load is distributed among nodes. In order to give an insight into the actual distribution of the routing load, figure 9.1 illustrates the distribution of the number of routed messages (namely, the probability with which a node routes a given number of messages). The results are obtained both analytically and via simulation.

We can observe from the figure that the routing load with e-Chord is much more balanced that with Chord, which explains the FI results obtained in Section 9.5.1. Specifically, with e-Chord the distribution is centered around the average and sharply decreases for smaller and larger numbers, which means that most nodes route a number of messages close to the average. Instead, with Chord we have a much less balanced distribution, with many nodes that route few messages and a long tail with heavily loaded nodes.

We further observe that the analytical results follow simulations reasonably closely, although there is some small deviation. This deviation is caused by our assumption that all the nodes that have the same number of fingers route the same number of messages. This assumption hides some randomness in the number of routed messages that our model does not account for.

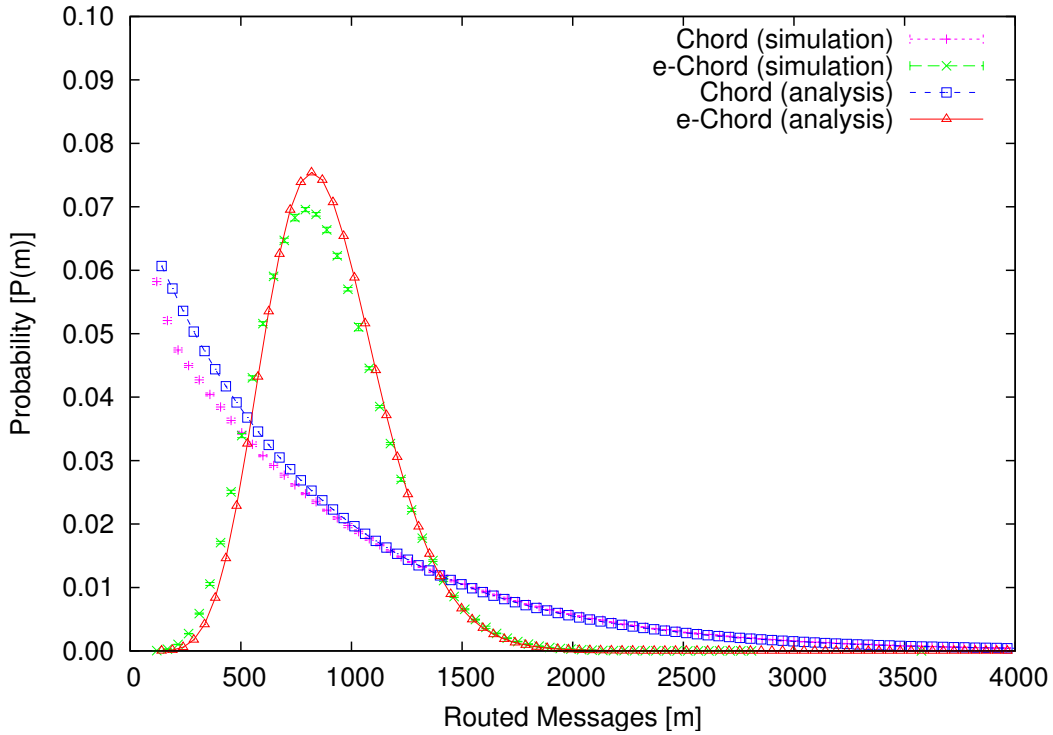


Figure 9.1: *Histogram of the messages routed by Chord/e-Chord nodes ($n = 10^6, s = 16, q = 10^8$)*

9.5.3 Inbound fingers distribution

The key idea behind e-Chord is to balance the number of fingers that point at each node by redistributing the inbound fingers that a node receives among its successors. In order to evaluate how well e-Chord achieves this objective, figure 9.2 depicts the distribution of the number of fingers for Chord and e-Chord (in particular, the figure shows the probability that a node has a given number of inbound fingers f), obtained analytically and via simulation.

We first observe from the figure that e-Chord provides a much more balanced distribution than Chord. Indeed, with e-Chord almost all nodes have a number of fingers close to the average given by the peak in the distribution, while Chord exhibits a much more unbalanced distribution. Note that distributions have a similar shape to the ones of figure 9.1.

We conclude that e-Chord achieves the objective of balancing the number of inbound fingers. We further observe from the figure that analytical results follow simulations very closely, which validates this part of the analysis.

9.5.4 Routing load vs. number of fingers

One key step of our analysis is the computation of the number of messages routed by a node as a function of the number of fingers pointing to this node. This relationship is given by Eq. (9.15). In order to validate this part of the analysis, we performed the following

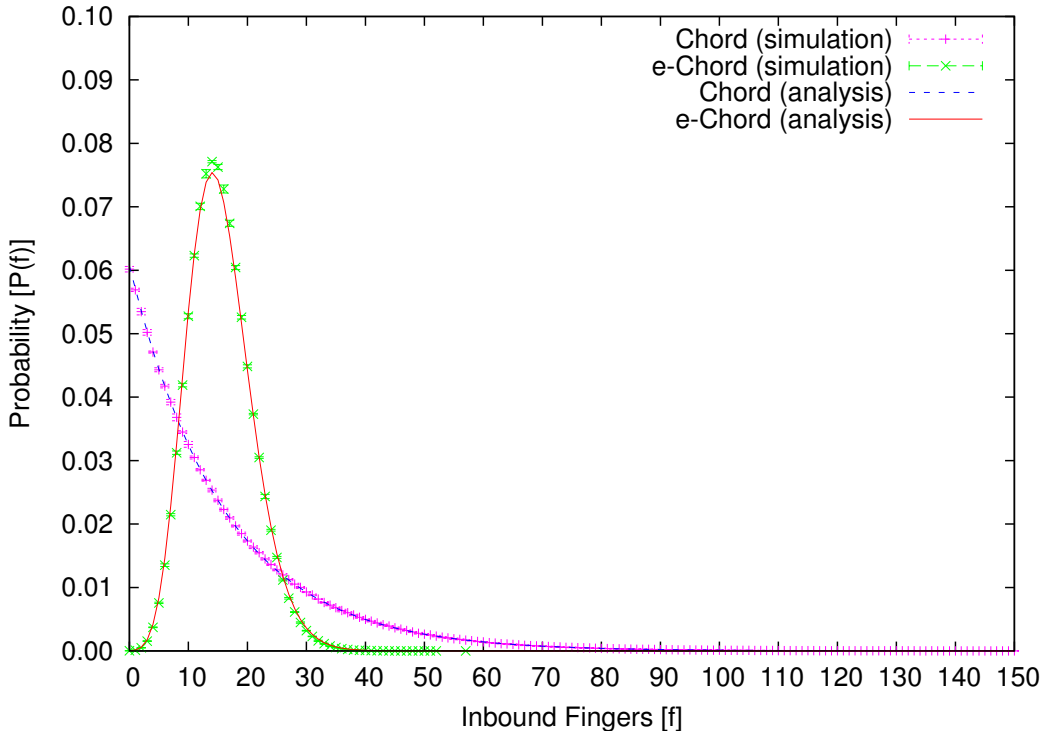


Figure 9.2: *Distribution of inbound fingers pointing to Chord/e-Chord Nodes ($n = 10^6, s = 16$)*

experiment. We evaluated the number of messages routed by all the nodes that have the same number of fingers, took their average and compared it against the value given by Eq. (9.15).

Figure 9.3 illustrates the number of messages routed by all nodes, their mean as well as the value given by the analysis. We observe that the average value matches with the analytical results, which validates this part of the analysis. There are only some deviations for large numbers of fingers, however this region contains only few samples which make the results not statistically significant.

9.5.5 Number of hops

Since e-Chord introduces some changes in the set up of the fingers, this could raise the question of whether choosing different fingers can have a negative impact on routing. In order to evaluate this, we measured the routing performance in terms of average number of hops for rings of different size n. The results obtained are illustrated in figure 9.4. We observe from these results that routing in e-Chord is not negatively affected by the different choice of fingers, and not only that, but it actually performs slightly better than Chord since fingers are better spread over the e-Chord ring. Therefore we conclude that e-Chord does not pay any price in lookup performance to achieve a better routing fairness.

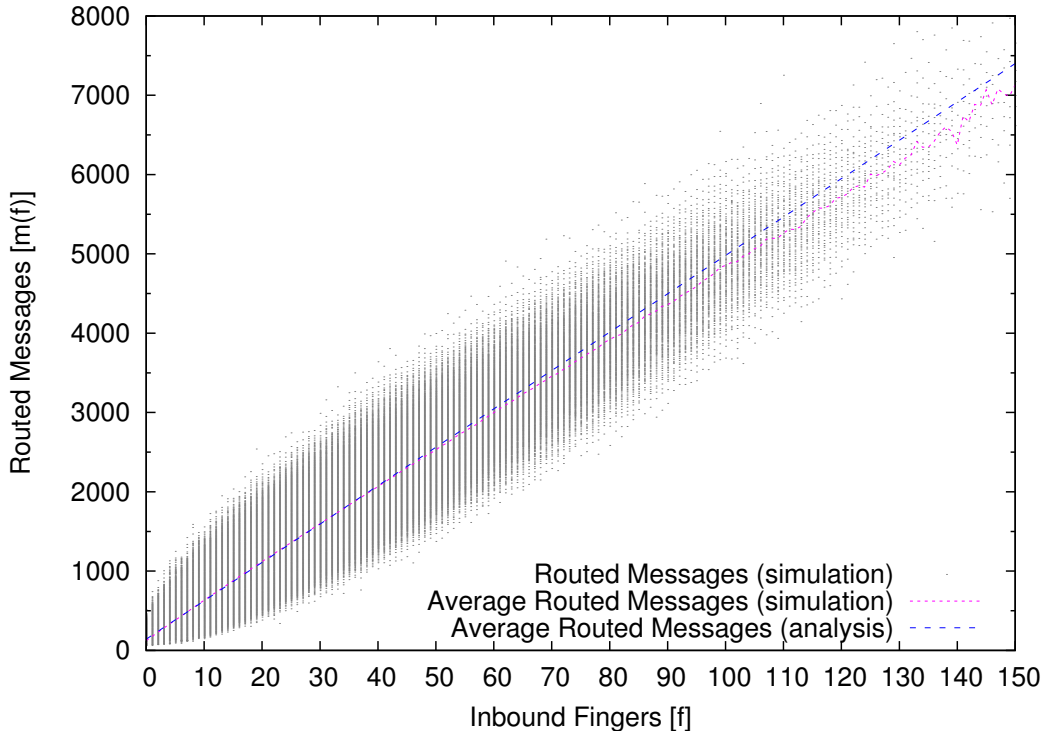


Figure 9.3: *Routed messages vs. inbound fingers per node in Chord* ($n = 10^6, s = 16, q = 10^8$)

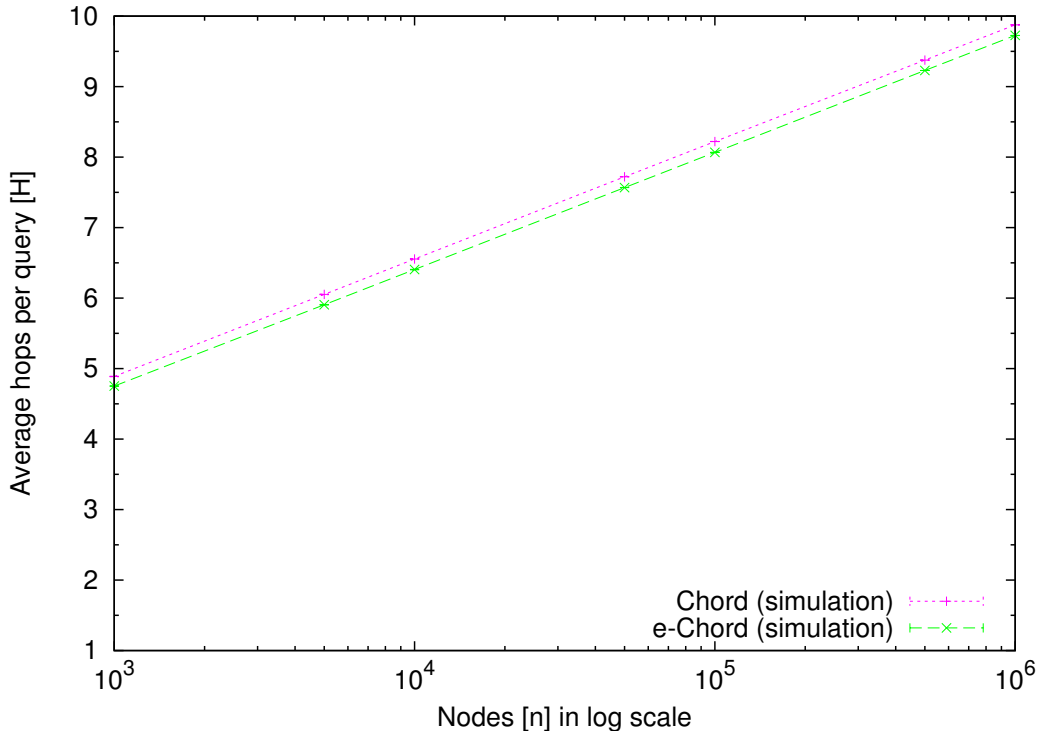


Figure 9.4: *Average number of hops per query in Chord/e-Chord* ($s = 16$)

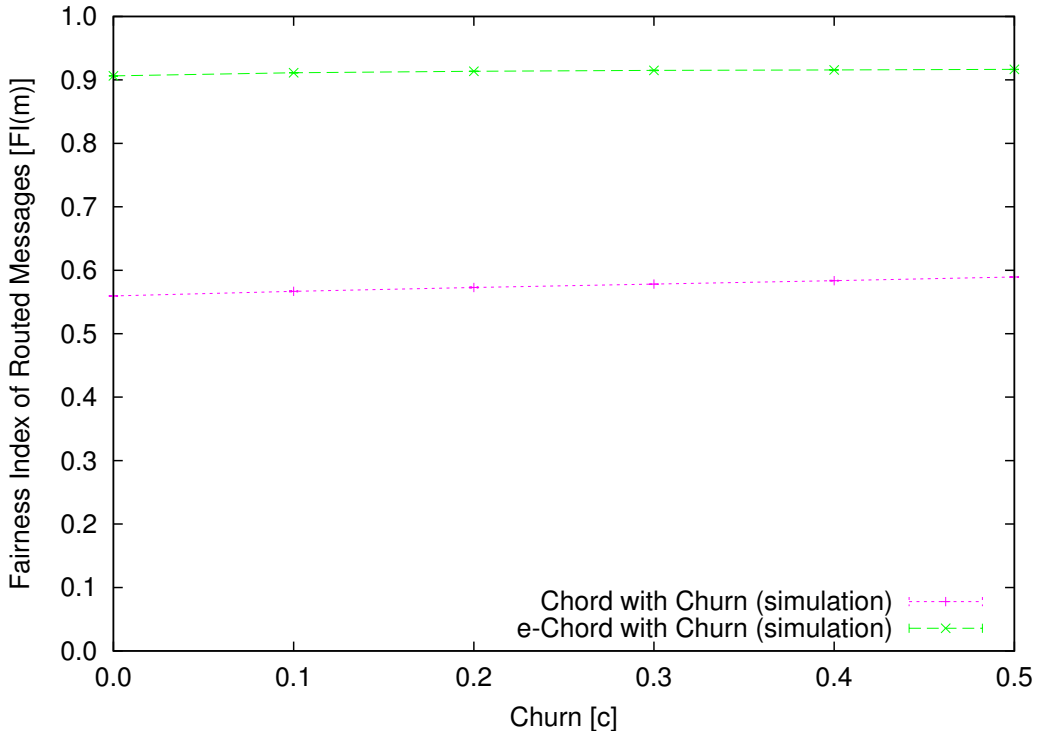


Figure 9.5: *Fairness index of the messages routed by Chord/e-Chord nodes under Churn ($n = 10^6, s = 16, q = 10^8$)*

9.5.6 Churn conditions

Our analysis and simulations so far have assumed static conditions in which all nodes are permanently active. In order to gain insight into the impact of churn conditions into the performance of Chord and e-Chord, we ran the following experiment. We divided the simulation into cycles of 10.000 queries each. At the beginning of each cycle, each node left the system with a probability $c \in [0.0, 0.5]$ to return in the next cycle. The fairness index FI resulting from this experiment with Chord and e-Chord are given in figure 9.5. We observe that the FI of Chord and e-Chord keeps almost constant independent of c; there is only a slight improvement when c grows which is caused by the fact that the worst-off nodes see their load reduced as a result of being inactive for some time. In any case, e-Chord clearly outperforms Chord regardless of c, which confirms the effectiveness of the proposed solution also under churn conditions.

9.5.7 Zipf popularity

All the simulations so far has assumed that all objects are equally popular. While this assumption may be appropriate for some scenarios like P2PSIP, it may be less suitable for other scenarios such as file sharing. Indeed, Zipf have been shown to be a good approximation [GDS+03] of the popularity distribution in P2P systems and have been widely

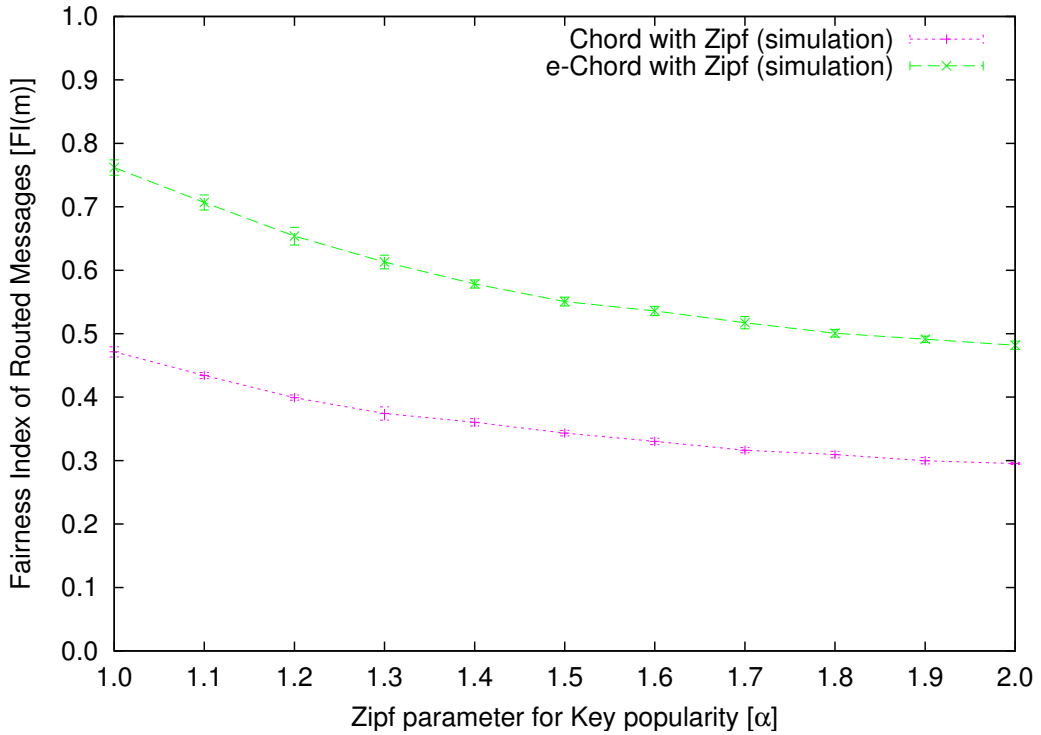


Figure 9.6: *Fairness Index of the messages routed by Chord/e-Chord nodes with $Zipf(10^5, \alpha)$ object popularity ($n = 10^6, s = 16, q = 10^8$)*

used [SBKF07] to reflect the skewed popularity level of the different contents in a file sharing scenario.

In order to understand the performance of Chord and e-Chord under a Zipf popularity distribution, we simulated them and evaluated the resulting FI varying the α parameter of a Zipf distribution with the most popular key being queried from $q_{max} = 10^5$ different nodes. The results, given in figure 9.6, show that both Chord and e-Chord become unfairer as the Zipf α parameter increases. The reason is that, when some objects become a lot more popular than others, the routes that lead to the popular objects become more loaded independent of the algorithm used. However, e-Chord keeps outperforming Chord by a similar degree, which confirms the effectiveness of e-Chord also for this case.

9.5.8 Heterogeneous nodes

In case of heterogeneous nodes, it may be desirable to assign more load to those nodes that have a larger capacity and/or computational power. While other approaches [GLS+04, RLS+03, GS05, ZH04, ZH05] require of additional overhead to support this functionality, with e-Chord this can be achieved just by $i)$ assigning different weights to the different nodes depending on their capacity, and $ii)$ accounting for these weights when redirecting fingers among successors. In order to assess the feasibility of e-Chord with an heterogeneous

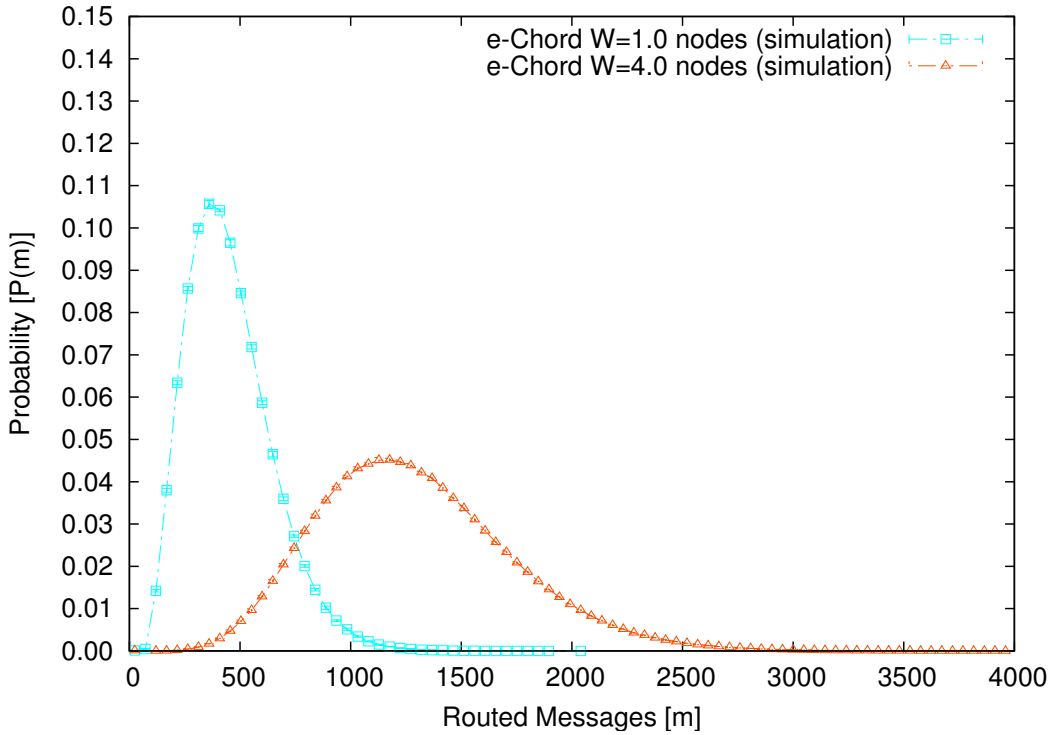


Figure 9.7: *Distribution of the messages routed by heterogeneous Chord/e-Chord Nodes* ($n = 10^6, s = 16, q = 10^8$)

population, we performed the following experiment. We divided the nodes of the ring in two categories: low capacity and high capacity nodes, and assigned a weight $W = 1$ to the former and $W = 4$ to the latter. Figure 9.7 depicts the distribution of the routing load for both categories. It can be observed that the load of the high capacity nodes is larger than the load of the low capacity nodes, which confirms the ability of e-Chord to support heterogeneous scenarios without incurring any extra overhead.

9.6 Summary and conclusions

While fairness in P2P has been widely studied for stored data, much less effort has been devoted to studying the fairness on routing load. In this chapter we have argued that routing represents the main cost in those DHT-based P2P systems where the size of the stored objects is small, and we have addressed this issue for Chord.

Our analysis of the routing fairness in Chord has been based on applying the well accepted Jain's Fairness Index to the routing load supported by each node in the Chord ring. To compute this metric, we have modelled the distribution of the routing load. We have done this by i) computing the distribution of the size of the zone between two consecutive nodes, ii) calculating from this the distribution of the number of fingers that point to a node, and iii) obtaining from this the routing load distribution. The analysis we have conducted has

shown that Chord's routing unfairness is mainly caused by the different zone sizes between nodes. Indeed, the larger the zone of responsibility of a node, the more fingers it will receive and therefore the larger its routing load will be.

In order to mitigate Chord's unfairness, we have proposed a simple enhancement to the finger selection mechanism of Chord. In the proposed algorithm, when receiving a query to set up a finger, a node replies with the address of one of its successors chosen randomly. This balances the number of fingers that point to a node, since this number does no longer depend on the size of its zone, which results in a more balanced routing load. One key advantage of the proposed approach is that, since a node in Chord already knows the addresses of all its successors, the proposed algorithm does not add any extra overhead.

The analytical results obtained in this work, which have been validated by simulation, show that the proposed enhancement to Chord improves very substantially the performance of Chord. Indeed, the Jain's Fairness Index, which provides a measure between 0 and 1, is about 0.6 for the number of messages routed by Chord nodes, while the enhanced Chord proposal achieves a fairness index of 0.9. In addition to the fairness index, we have also evaluated analytically the distribution of the number of fingers and the routing load. The results obtained show that enhanced Chord provides a much better balanced distributions, which explains the big difference in fairness indexes.

In addition to the analysis, we have also conducted a simulation study to gain insight into the performance of the proposed solutions. Simulations have been performed for up to 10^6 nodes, which corresponds to realistic implementations of P2P systems. Simulation results have shown that the improvement in fairness of our solution does not involve any price in performance; on the contrary, it behaves even slightly better than Chord in terms of the number of hops required to reach the destination of a query. Simulation results have also shown the effectiveness of the proposed solution under churn conditions, Zipf-like object popularity and heterogeneous nodes.

Part IV

Conclusion and Future Work

Chapter 10

Conclusion

In this Thesis we have first addressed the problem of Dynamic and Location Aware Service Discovery. We have proposed a simple and practical solution based on a Chord-like DHT. We have defined 11 design parameters and compared our solution to the related work based on them. We have shown that our solution is the one that better fulfils these design criteria. Furthermore we have validated the performance of our solution in two different cases of study: (i) NAT Traversal Servers Discovery and (ii) Home Agents Discovery in Mobile IP scenarios. The former allows us to validate the performance of our solutions in a highly dynamic environment. We have shown that the delay associated to the relayed communications is similar to the delay of direct communication, when using our solution to discover the Relay nodes. Furthermore, we have demonstrated that our solution is ISP friendly since it generates a very reduced amount of transit traffic associated to the relayed communications. Moreover, the second case of study validates our solution in a classical client/server scenario where the server is expected to be an *always on* well-known host. We have shown that our solution largely reduces the communication delay compared to MIPv4 and NEMO basic solutions. In addition, we have also demonstrated the scalability of our solution even in stressful conditions.

The extra overhead suffered from the servers in our solution is due to their participation in the Chord-DHT. Therefore, it is desirable to fairly balance this extra overhead among the servers participating in the system. In our system the size of the stored objects is small (e.g. IP addresses), thus routing dominates the cost of publishing and retrieving objects. Therefore, in order to equalize the extra-overhead of the servers in our system we need to balance the routing load. We have devoted the second part of this Thesis to study the routing fairness in Chord. Our analysis has been based on applying the well accepted Jain's Fairness Index (FI) to the routing load supported by each node in the Chord ring. We have analytically demonstrated that Chord performs poorly with a FI around 0.6. In order to improve this we have proposed a simple modification in the finger selection procedure of Chord that based on our analysis increases the FI up to 0.9 while introducing a negligible overhead. Additionally to the analysis we have conducted large-scale simulations with up to 10^6 nodes to gain insight into the performance of the proposed solution. The obtained results

validate our analysis and shows that our solution does not add any performance degradation. Finally, the simulations demonstrate that our solution also outperforms Chord under churn conditions, Zipf-like object popularity and heterogeneous nodes.

Chapter 11

Future Work

As future work in the area of Dynamic and Location Aware Server Discovery we will first detail a specification for an Anycast Platform based on our solution. This means, a common architecture to be used for multiple services in order to find a close-by server. The key point for designing such an infrastructure has been introduced in Chapter 6. We are also interested in designing the specific solution for the case of geographical coordinates. As stated in Chapter 5, we would like to explore the utilization of a different type of DHT (e.g. CAN) more appropriate in this case. Finally, we are working on the implementation of our solution using e-chord as DHT.

Regarding Fairness in Chord we would like to address the following issues: (i) Extend the model of Routing Fairness for heterogeneous scenarios. We aim to model the weights to be assigned to each node in the Chord-DHT based on the node's capacity. This weight will represent the probability that the node has to be selected in the new defined finger selection procedure; (ii) Remove the bottleneck of the routing task happening in the last hop, i.e. the predecessor of the targeted key. In order to remove this bottleneck we are exploring a predecessor based Chord solution in which the responsible node of a given key is its predecessor instead of its successor; (iii) Extend our solution to address the case of Zipf content popularity distribution. For this purpose we will explore dynamic replication techniques. Finally, we are already working in the implementation of e-Chord using as basis the OpenChord[1] code.

[1] http://open-chord.sourceforge.net/

Chapter 12

Contributions

The main contributions of this Thesis in the area of Location Aware Server Discovery are:

- R. Cuevas, A. Cuevas, A. Cabellos-Aparicio, L. Jakab, C. Guerrero, "A Collaborative P2P Scheme for NAT Traversal Server Discovery based on Topological Information", Elsevier Computer Networks (Special Issue on Collaborative Peer-to-Peer Systems). *Accepted for Publication*, 2010. (Acceptance Rate 20%).
- R. Cuevas, A. Cabellos-Aparicio, A. Cuevas, J.Domingo-Pascual, A.Azcorra, " fP2P-HN: A P2P-based Route Optimization Architecture for Mobile IP-based Community Networks". Elsevier Computer Networks (Special Issue on Content Distribution Infrastructures for Community Networks), vol 53. Issue 4. March 2009. (Acceptance Rate 12.5%)

The first one addresses the problem of NAT Traversal Server Discovery whereas the second one presents our specific solution for the case of Home Agent Discovery in Mobile IP Scenarios. Some other relevant contributions in the area of Home Agent Discovery are:

- A. Cabellos-Aparicio, R. Cuevas, J. Domingo-Pascual, A. Cuevas, C.Guerrero, "fP2P-HN: A P2P-based Route Optimization Solution for Mobile IP and NEMO clients". IEEE International Conference in Communications 2009 (ICC 2009). Dresden (Germany). 14-18 June, 2009. (Acceptance Rate 34.9%)
- R. Cuevas, C. Guerrero, A. Cuevas Rumin, M. Calderón, C.J. Bernardos, "A P2P Based Architecture for Global Home Agent Dynamic Discovery in IP Mobility", 65th Semi Anual IEEE Vehicular Technology Conference (Spring VTC 2007). Dublin (Ireland). 22-25 April 2007. (Acceptance Rate 40%)

Also we would like to mention some minor contributions in the security aspects of the specific solution for the case of Home Agents discovery:

- A. Cuevas, R. Cuevas, M. Urueña, C Guerrero, "A Novel Overlay Network for a Secure Global Home Agent Dynamic Discovery", 1st International Workshop on Peer to Peer Networks (PPN'07). Vilamoura, Algarve (Portugal), Nov 25-30, 2007. LNCS 4806. Pages: 921-930, ISSN: 0302-9743. ISBN: 978-3-540-76889-0.

- R. Cuevas, A. Cuevas, C Guerrero, M. Urueña, J. Gutierrez , "A Secure Approach for Global Home Agent Dynamic Discovery based on Peer-to-Peer", Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2007). Vienna (Austria). 24 September 2007.

Finally, our contribution in the field of Routing Fairness in Chord is:

- R. Cuevas, M. Urueña, A. Banchs, "Fairness Routing in Chord: Analysis and Enhancement". 28th IEEE Conference on Computer Computer Communications (INFOCOM 2009). Rio de Janeiro, Brasil. 19-25 April, 2009. (Acceptance Rate 19.7%)

References

[ABKM01] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131–145, 2001.

[AD07] C. Aoun and E. Davies. Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966 (Informational), 2007.

[AL09] Sharad Agarwal and Jacob R. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. *ACM SIGCOMM'09*, 2009.

[BC02] Nevil Brownlee and KC Claffy. Understanding internet traffic streams: Dragonflies and tortoises. *IEEE Communications Magazine*, 40, 2002.

[BCM03] John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple load balancing for distributed hash tables. In *IPTPS*, 2003.

[Bel92] S. Bellovin. A best-case network performance model. techical report, att, 1992.

[BFWP05] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig. NATBlaster: Establishing TCP connections between hosts behind NATs. In *ACM SIGCOMM Asia Workshop*. ACM, April 2005.

[BGMBA06] Marcelo Bagnulo, Alberto Garca-Martnez, Carlos J. Bernardos, and Arturo Azcorra. Scalable Support for Globally Moving Networks. In *Proceedings of the 3rd International Symposium on Wireless Communication Systems (ISWCS)*, Valencia (Spain), September 2006.

[boe] Boeing Connexion Service. http://www.connexionbyboeing.com".

[BS06] S. A. Baset and H. G. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *In proceedings INFOCOM 2006. 25th IEEE International Conference on Computer Communications*, pages 1–11. IEEE, 2006.

[CAC07] The Impact of P2P File Sharing, Voice over IP, Skype, Joost, Instant Messaging, One-Click Hosting and Media Streaming such as YouTube on the Internet, 2007. http://www.ipoque.com/resources/internet-studies/internet-study-2007.

[CACDP+09] Alberto Cabellos-Aparicio, Rubén Cuevas, Jordi Domingo-Pascual, Ángel Cuevas, and Carmen Guerrero. fP2P-HN: A P2P-based Route Optimization Solution for Mobile IP and NEMO clients. In *In proceedings IEEE International Conference in Communications 2009 (ICC 2009)*. IEEE, 2009.

[CADP07] Albert Cabellos-Aparicio and Jordi Domingo-Pascual. A flexible and distributed home agent architecture for mobile ipv6-based networks. In *IFIP Networking*, 2007.

[CASBDP09] Albert Cabellos-Aparicio, Damien Saucez, Olivier Bonaventure, and Jordi Domingo-Pascual. Validation of a LISP simulator. Technical Report UPC-DAC-RR-CBA-2009-8, UPC, 2009.

[CB08] David R. Choffnes and Fabián E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. *SIGCOMM Comput. Commun. Rev.*, 38(4):363–374, 2008.

[CBB+06] Mara Caldern, Carlos J. Bernardos, Marcelo Bagnulo, Ignacio Soto, and Antonio de la Oliva. Design and Experimental Evaluation of a Route Optimization Solution for NEMO. *IEEE Journal on Selected Areas in Communications*, 24(9):1702–1716, September 2006.

[CC97] Robert L. Carter and Mark E. Crovella. Server Selection Using Dynamic Path Characterization in Wide-Area Networks. In *IEE INFOCOM '97*, 1997.

[CCA+09] Rubén Cuevas, Alberto Cabellos-Aparicio, , Ángel Cuevas, Jordi Domingo-Pascual, and Arturo Azcorra. fP2P-HN: A P2P-based Route Optimization Architecture for Mobile IP-based Community Network. *Elsevier Computer Networks (Special Issue on Content Distribution Infrastructures for Community Networks)*, 2009.

[CCCA+10] Rubén Cuevas, Ángel Cuevas, Alberto Cabellos-Aparicio, Lorand Jakab, and Carmen Guerrero. A Collaborative P2P Scheme for NAT Traversal Server Discovery based on Topological Information. *Elsevier Computer Networks (Special Issue on Collaborative Peer-to-Peer Systems)*, 2010. Accepted for Publication.

[CCG+07] Rubén Cuevas, Ángel Cuevas, Carmen Guerrero, Manuel Urueña, and Jose Gutiérrez. A Secure Approach for Global Home Agent Dynamic Discovery based on Peer-to-Peer. In *In proceedings Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2007)*, 2007.

[CCRK03] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet Coordinates for Distance Estimation. In *ICDCS*, 2003.

[CCUnG07] Ángel Cuevas, Rubén Cuevas, Manuel Urueña, and Carmen Guerrero. A Novel Overlay Network for a Secure Global Home Agent Dynamic Discovery. In *LNCS 4806. In proceedings 1st International Workshop on Peer to Peer Networks (PPN'07)*. Springer, 2007.

[CF07] M. Cadaco and M. Freedman. Illuminati - Opportunistic Network and Web Measurement, September 2007. http://illuminati.coralcdn.org/stats/.

[CGC+09] Rubén Cuevas, Carmen Guerrero, Ángel Cuevas, María Calderón, and C.J Bernardos-Cano. A P2P Based Architecture for Global Home Agent Dynamic Discovery in IP Mobility. In *In proceedings 65th Semi Anual IEEE Vehicular Technology Conference (Spring VTC 2007)*. IEEE, 2009.

[CIS05] *Enterprise QoS Solution Reference Network Design Guide*. CISCO, 2005.

[clu] N1 Grid Engine. www.sun.com/software/gridware/.

[Coh03] Bram Cohen. Incentives Build Robustness in BitTorrent. In *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[CUnB09] Rubén Cuevas, Manuel Urueña, and Albert Banchs. Routing fairness in Chord: Analysis and Enhancement. In *In proceedings INFOCOM 2009. 28th IEEE International Conference on Computer Communications*. IEEE, 2009.

[DCKM04] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. In *ACM SIGCOMM '04*, 2004.

[DKK+01] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*. ACM, 2001.

[DLS+04] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. Designing a dht for low latency and high throughput. In *NSDI*, 2004.

[DMHG08] Marcel Dischinger, Alan Mislove, Andreas Haeberlen, and Krishna P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008.

[DWPT05] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963 (Proposed Standard), January 2005.

[EF94] K. Egevang and P. Francis. RFC 1631 The IP Network Address Translator (NAT), May 1994.

[FJJ+01] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: a global internet host distance estimation service. *IEEE/ACM Transaction on Networkings*, 2001.

[FLM06] Michael J. Freedman, Karthik Lakshminarayanan, and David Mazières. OASIS: anycast for any service. In *USENIX NSDI'06*, 2006.

[FSK05] Bryan Ford, Pyda Srisuresh, and Dan Kegel. Peer-to-peer communication across network address translators. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 13–23. USENIX Association, 2005.

[GDS+03] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *SIGOPS Oper. Syst. Rev.*, 2003.

[geo] GeoIP Tool. http://www.geoiptool.com.

[GF05] Saikat Guha and Paul Francis. Characterization and measurement of TCP traversal through NATs and firewalls. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 18–18. USENIX Association, 2005.

[GLS+04] Brighten Godfrey, Karthik Lakshminarayanan, Sonesh Surana, Richard M. Karp, and Ion Stoica. Load balancing in dynamic structured p2p systems. In *INFOCOM*, 2004.

[GMG+04] Krishna P. Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David Wetherall. Improving the reliability of internet paths with one-hop source routing. In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*. USENIX Association, 2004.

[GS95] James D. Guyton and Michael F. Schwartz. Locating nearby copies of replicated Internet servers. *ACM SIGCOMM'95*, 1995.

[GS05] Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *IEEE INFOCOM*, 2005.

[gsm93a] European Telecommunications Standards Institute, GSM 02.09: Security Aspects, 1993.

[gsm93b] European Telecommunications Standards Institute, GSM 03.20: Security Related Network Functions, 1993.

[gsm00] European Telecommunications Standards Institute, TS 133 102: Universal Mobile Telecommunications System (UMTS); 3G Security; Security Architecture, version 3.6.0, 2000.

[HFC+08] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, John C.S. Lui, and Cheng Huang. Challenges, design and analysis of a large-scale p2p-vod system. In *ACM SIGCOMM 2008 conference on Data communication*, pages 375–388. ACM, 2008.

[hos] HostIP. http://www.hostip.info/dl/index.html.

[HS03] Ningning Hu and Peter Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 2003.

[Hus01] G. Huston. Commentary on Inter-Domain Routing in the Internet. RFC 3221 (Informational), December 2001.

[i3] Internet Indirect Infrastructure (i3). http://i3.cs.berkeley.edu/.

[INE] INET. Internet Topology Generator. http://topology.eecs.umich.edu/inet/.

[ip2] IP2Location. http://www.ip2location.com.

[iPl] iPlane. iPlane: An Information Plane for Distributed Services. http://iplane.cs.washington.edu/.

[Ipo07] Ipoque. Ipoque internet study 2007, 2007. http://www.ipoque.com/userfiles/file/internet_study_2007.pdf.

[itu96] One-Way Transmission Time, 1996. ITU Rec. G.114, International Telecommunications Union.

[Jac] V. Jacobson. Pathchar.

[JBRAS03] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003.

[JCH98] Raj Jain, Dah-Ming Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, 1998.

[JD03] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Trans. Netw.*, 2003.

[JJJ+00] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. On the placement of internet instrumentation. In *INFOCOM*, 2000.

[JPA04] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[KLL+97] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997.

[KM02] T. Klinberg and R. Manfredi. Gnutella 0.6, 2002. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.txt.

[KM05] Krishnaram Kenthapadi and Gurmeet Singh Manku. Decentralized algorithms using both local and random probes for p2p load balancing. In *SPAA '05: Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2005.

[KR00] Wolfgang Theilmann Kurt and Kurt Rothermel. Dynamic Distance Maps of the Internet. In *IEEE INFOCOM'00*, 2000.

[KR04] David R. Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2004.

[KRT+09] Sebastian Kaune, Ruben Cuevas Rumin, Gareth Tyson, Andreas Mauthe, Carmen Guerrero, and Ralf Steinmetz. Unraveling BitTorrent's File Unavailability: Measurements, Analysis and Solution Exploration, 2009. http://arxiv.org/abs/0912.0625.

[KW00] Dina Katabi and John Wroclawski. A framework for scalable global IP-anycast (GIA). *SIGCOMM Comput. Commun. Rev.*, 30(4), 2000.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.

[LDNP08] K. Leung, G. Dommety, V. Narayanan, and A. Petrescu. Network Mobility (NEMO) Extensions for Mobile IPv4. RFC 5177 (Proposed Standard), April 2008.

[LGS07] Jonathan Ledlie, Paul Gardner, and Margo I. Seltzer. Network Coordinates in the Wild. In *USENIX NSDI*, 2007.

[LS05] Jonathan Ledlie and Margo I. Seltzer. Distributed, secure load balancing with skew, heterogeneity and churn. In *IEEE INFOCOM*, 2005.

[Max] MaxMind. http://www.maxmind.com/.

[MEX] MEXT. Mobility EXTensions for IPv6. http://www.ietf.org/html.charters/mext-charter.html.

[MIP+06] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: an information plane for distributed services. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 367–380. USENIX Association, 2006.

[MM02] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '02. First International Workshop on Peer-to-Peer Systems*, 2002.

[NH04] C. Ng and J. Hirano. Extending return routability procedure for network prefix (RRNP). IETF Draft, 2004.

[NTWZ07] C. Ng, P. Thubert, M. Watari, and F. Zhao. Network Mobility Route Optimization Problem Statement. RFC 4888 (Informational), July 2007.

[NZ01] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *IEEE INFOCOM '01*, 2001.

[p2p] P2PSIP Working Group. http://www.ietf.org/html.charters/p2psip-charter.html/.

[PCW+03] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for Scalable Distributed Location, 2003.

[Per] Perl. The perl directory. www.perl.org.

[Per02] C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), August 2002. Updated by RFC 4721.

[PLBV05] Santashil PalChaudhuri, Jean-Yves Le Boudec, and Milan Vojnovic. Perfect simulations for random trip mobility models. In *ANSS '05: Proceedings of the 38th annual Symposium on Simulation*, pages 72–79, Washington, DC, USA, 2005. IEEE Computer Society.

[pma] Passive Measurement and Analysis (PMA). http://pma.nlanr.net.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350, London, UK, 2001. Springer-Verlag.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2001.

[RGZ06] Shansi Ren, Lei Guo, and Xiaodong Zhang. Asap: an as-aware peer-relay protocol for high quality voip. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. IEEE Computer Society, 2006.

[RLH06a] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4), January 2006.

[RLH06b] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[RLS+03] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in structured p2p systems. In *IPTPS '03. Second International Workshop on Peer-to-Peer Systems*, 2003.

[RMM09] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), April 2009.

[Ros07] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, October 2007.

[rou] Routeview Project. http://www.routeviews.org/.

[RWHM03] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), 2003.

[SBKF07] Sabina Serbu, Silvia Bianchi, Peter Kropf, and Pascal Felber. Dynamic load sharing in peer-to-peer systems: When some peers are more equal than others. *IEEE Internet Computing*, 2007.

[SENB07] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of kad. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007.

[SKK03] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transaction on Networking*, 11(1):17–32, 2003.

[SPR09] Georgos Siganos, Josep M. Pujol, and Pablo Rodriguez. Monitoring the Bittorrent Monitors: A Bird's Eye View. In *PAM'09: Proceedings of the Passive Active Measurement conference*, volume 5448 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2009.

[Sta] Ookla's Speedtest Throughput Measures.

[SVF08] T. Steele, V. Vishnumurthy, and P. Francis. A parameter-free load balancing mechanism for p2p networks. In *IPTPS*, 2008.

[VB07] Iljitsch Van Beijnum. IPv4 Address Consumption. *The Internet Protocol Journal*, 10(3):22–28, 2007.

[vuz] Vuze Network Status Monitor. http://azureus.sourceforge.net/plugin_details.php?plugin=aznetm

[W.B04] W.B.Norton. The Evolution of the U.S. Internet Peering Ecosystem, 2004.

[WBS08] Kho Wookyun, S.A. Baset, and H. Schulzrinne. Skype relay calls: Measurements and experiments. In *In proceedings INFOCOM Workshops 2008,. 27th IEEE International Conference on Computer Communications*, pages pp.1–6. IEEE, 2008.

[WCL+02] Chun Hsin Wu, Ann Tzung Cheng, Shao Ting Lee, Jan Ming Ho, and Der Tsai Lee. Bi-directional route optimization in mobile ip over wireless lan, 2002.

[WD05] Xugang Wang and Qianni Deng. VIP: A P2P Communication Platform for NAT Traversal. In *Parallel and Distributed Processing and Applications, Third International Symposium, ISPA 2005*, pages 1001–1011, 2005.

[Wel00] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. RFC 3007 (Proposed Standard), 2000. Updated by RFCs 4033, 4034, 4035.

[who] WHOIS service. http://www.netdemon.net/tutorials/whois.txt.

[WOM05] Ryuji Wakikawa, Yasuhiro Ohara, and Jun Murai. Virtual mobility control domain for enhancements of mobility protocols. In *INFOCOM*. IEEE, 2005.

[WSS05] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a lightweight network location service without virtual coordinates. *ACM SIGCOMM'05*, 2005.

[Xin07] Jiang XingFeng. A mechanism to discover STUN/TURN nodes in P2PSIP. draft-jiang-p2psip-stun-turn-discovery-00, 2007.

[XYK+08] Haiyong Xie, Richard Y. Yang, Arvind Krishnamurthy, Yanbin G. Liu, and Abraham Silberschatz. P4P: Provider Portal For Applications. *SIGCOMM Comput. Commun. Rev.*, 38(4):351–362, 2008.

[YHC06] Yun Sheng Yen, Chia Chang Hsu, and Han Chieh Chao. Global dynamic home agent discovery on mobile IPv6: Research Articles. *Wirel. Commun. Mob. Comput.*, 6(5):617–628, 2006.

[ZDWR] C. Zhang, P. Dunghel, D. Wu, and K.W. Ross. Unraveling the BitTorrent Ecosystem. Technical Report.

[ZH04] Yingwu Zhu and Yiming Hu. Towards efficient load balancing in structured p2p systems. In *IEEE IPDPS*, 2004.

[ZH05] Yingwu Zhu and Yiming Hu. Efficient, proximity-aware load balancing for dht-based p2p systems. *IEEE Trans. Parallel Distrib. Syst.*, 16(4), 2005.