From inter-connecting P2P overlays to co-operating P2P systems

Hoang Giang Ngo

UNIVERSITY OF NICE - SOPHIA ANTIPOLIS

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

Doctor of Computer Science

of the University of Nice - Sophia Antipolis

Defended by

Giang NGO HOANG

From Inter-connecting P2P Overlays To Co-operating P2P Systems

Thesis Advisors: Luigi LIQUORI, Hung NGUYEN CHAN and Huong TRUONG THU

CO-TUTELLE with the HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY, VIETNAM

Jury :

President :	Marina RIBAUDO	-	Associate Professor, University of Genoa
Reviewers :	Giovanni CHIOLA	-	Professor Emeritus, University of Genoa
	Thanh NGUYEN HUU	-	Associate Professior, HUST
Examiners :	Fabrice HUET	-	Associate Professor, University of Nice
	Frederic GIROIRE	-	Chargé de recherche, CNRS
	Carlos BALADRÓN	-	PhD, University of Valladolid
Advisors :	Luigi LIQUORI	-	Research Director, INRIA
	Hung NGUYEN CHAN	-	Associate Professior, VIELINA
	Huong TRUONG THU	-	PhD, HUST
Invited :	Jean-Claude BERMOND	-	Research Director, CNRS

Acknowledgments

First of all, I would like to thank my three advisors: professors Luigi Liquori, Hung Nguyen Chan and Huong Truong Thu. Thank you Luigi and professor Hung for your continuous support of my Ph.D study and research, for your patience, motivation and enthusiasm. Your guidance helped me in all the time of research and writing of this thesis. Thank you Dr. Huong for your kind help.

I would also like to thank professor Giovanni Chiola and professor Thanh Nguyen Huu, for doing me the honor of being rapporteurs for my thesis, as well as professor Fabrice Huet , professor Marina Ribaudo , Dr. Frederic Giroire and Dr. Carlos Baladrón for graciously accepting to be part of the jury.

My sincere thanks also go to Vincenzo Ciancaglini and Petar Maximović. Thank you for your stimulating discussion and contributing to my research works.

My deepest gratitude goes to my wife and my daughter. Thank you for being my endless sources of love and encouragement. I am so happy to have you in my life.

Last but not the least, I would like to thank other members of my family and my friends for always being with me in good time and bad, always encouraging and supporting me.

Abstract

Peer-to-peer (P2P) systems are used by millions of users every day. In many scenarios, it is desirable for the users from different P2P systems to communicate and exchange content or services with each other. This requires co-operation between the P2P systems, which is often difficult or impossible, due to the two following reasons. First is the lack of an inter-overlay routing infrastructure throughout these systems, caused by the incompatibilities in overlay networks on top of which they are built. Second, there are incompatibilities in the application protocols of these systems.

The main topic of this thesis is enabling the cooperation between P2P systems. The thesis introduces a *cooperation framework* for backward-compatible cooperating heterogeneous P2P systems which constitutes two parts. The first one is an *inter-overlay routing framework* which allows to inter-routing between heterogeneous overlay networks. The second one is the *cooperation application*, built on the top of the *inter-overlay routing framework*, which aims at solving the incompatibilities in the application protocols of P2P systems. However, due to the differences in contexts of different application domains, in the *cooperation framework*, we only introduce the principles underpinning the building of *cooperation application*.

As a case study of the *cooperation framework*, we introduce a complete *solution* for cooperating P2P file-sharing networks which is applicable for all current P2P file sharing networks. The *solution* includes the *cooperation application*, running a protocol named *Inter-network File-sharing Protocol* (IFP), operating upon the *inter-overlay routing framework*. Through the experiments, we show the efficiency of the *inter-overlay routing framework* as well as the efficiency of the *solution* for cooperating P2P file-sharing networks.

While investigating the inter-overlay routing, we found that the inter-connecting P2P overlays can achieve important features including scalability, localization, resilience and self-organization. We believe that with these features, the inter-connecting P2P overlays are suitable to be communication architectures of many Cyber-Physical Systems (CPS). Therefore, in the second topic of this thesis, we investigate a case study of using inter-connecting P2P overlays for smart grid, a typical example of CPS. In the case study, a hierarchically inter-connecting Distributed Hash Table (DHT) serves as the fundamental part of communication infrastructure of Advanced Metering Infrastructure (AMI) in smart grid. The analysis and experiments show the scalability and communication efficiency of the proposed architecture.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivations and research directions

The research for this thesis has followed two main directions. The first one is cooperation of heterogeneous P2P systems and the second one is the ability of using inter-connecting P2P overlays as routing infrastructure for cyber-physical systems.

1.1.1 Cooperation of heterogeneous P2P systems

Many P2P systems have been introduced and become widely used today, such as P2P file-sharing networks, P2P instant messaging networks, P2P data storage systems, etc. Those systems co-exist but normally are isolated and compete with each other. On the other hand, there are clear advantages for these systems to co-operate to share the content or services with each other. For instance, users in one P2P system can achieve the content or services in other P2P systems. Another advantage is the overall storage for storing data across P2P systems is reduced because the cooperation allows reducing the number of copies of stored data. The content redundancy is also easy to achieve by storing the data at different P2P systems.

However, these P2P systems, even with the same goals, are different from each other in both the overlay networks and the applications built on top of those networks. While the overlay networks are heterogeneous in many aspects such as structures, topologies, routing algorithms, message encoding algorithms, the applications are different from each other in data context, protocols for exchanging content or services, etc. These differences result in an overall incompatibility across P2P systems which prevents the communication between P2P systems and at a higher level, the cooperation between them.

As a matter of fact, the main challenge for the cooperation between P2P systems is twofold. First is to introduce an inter-overlay routing infrastructure between isolated P2P systems. Second is to define a common interface for various applications and map the interface of each P2P system to the common interface and vice versa. Another challenge of cooperating P2P systems is to ensure the back-ward compatibility, i.e. the

peers are unaware of the cooperation protocols still work normally. This is important requirement for co-operating already existing P2P systems. The main part of this thesis, presented in chapter 3 and chapter 4, addresses those challenges.

1.1.2 Application of the inter-connecting P2P overlays in Cyber-Physical Systems (CPS)

A CPS is a system of collaborating computational elements controlling physical entities [CPS]. Many CPS constitutes a large number of sensors and actuators, deployed in vast geographical areas, monitoring and sending the data to control centers. These systems require scalable communication infrastructures to deal with the large amount of generated data. It is also desirable for these systems to be resilient with failure and self-organizing to reduce the operation and maintenance cost.

Of the communication architectures, the inter-connecting P2P overlays is a promising candidate. Beside the scalability and self-organization features, the overlays in the inter-connecting P2P overlays can be geographically distributed thus allow to bring the localization to the infrastructure. On the other hand, the fact that the overlays can be heterogeneous gives the great flexibility in building the infrastructure.

In this thesis, we exploit the inter-connecting P2P overlays architecture for the Advanced Metering Infrastructure (AMI) [AMI] in smart grid, a typical CPS. AMI is a large geographic distributed system that provides an intelligent connection between consumers and system operators in modern electric grid. One of the biggest challenge that AMI faces is to scalable collect and manage a huge amount of data. We will show that the inter-connecting P2P overlays can address this challenge.

1.2 Thesis structure

This thesis is structured as following:

Chapter 2 constitutes twofold. First we present the background for our works, including definition, classification, characteristics and application of P2P. Second, we provide a literature review of the studies on cooperation of P2P systems and point out the difference between the work done in this thesis and previous works.

Chapter 3 presents a *cooperation framework* for cooperating heterogeneous P2P systems. The *cooperation framework* constitutes an *inter-overlay routing framework* allowing to route the messages across different P2P systems and a *cooperation application*, built upon the routing infrastructure, allows to bridge those P2P systems at application level. However, each application domain has its own semantic, therefore it is difficult to introduce a *cooperation application* for all application domains. We introduce the principles for building the *cooperation application* which is applicable for all application domains instead. Based on the *cooperation framework*, one can develop a complete cooperation solution for each application domain. The *cooperation framework* allows the backward compatibility in the sense that the peers which are aware of the framework can transparently achieve the resources or services from other systems while the peers which are unaware of the framework can still work normally. The *cooperation framework* is the main contribution of our thesis.

In Chapter 4, as a case study of the *cooperation framework*, we introduce a complete *cooperation solution* for P2P file-sharing domain which is the most successful area of P2P.

The *cooperation solution* allows to backward compatible cooperating heterogeneous P2P file-sharing networks which is applicable for all current P2P file-sharing networks. In the *cooperation solution*, the IFP protocol is introduced as *cooperation application* solving the incompatibilities in application protocols of various P2P file-sharing networks.

In Chapter 5, we present an architecture of inter-connecting P2P overlays mixing with client-server model for the AMI in smart grid, a typical CPS. The inter-connecting P2P overlays constitutes multiple DHT overlays which are geographically distributed and hierarchically organized.

Chapter 6 resumes the presented works and introduces new direction for the research.

1.3 Publications

This thesis relies on following conference papers and reports:

1. Giang Ngo Hoang, Luigi Liquori and Hung Nguyen Chan. Backward-Compatible Cooperation of Heterogeneous P2P Systems. The 15th International Conference on Distributed Computing and Networking (ICDCN) 2014, accepted.

2. Giang Ngo Hoang, Luigi Liquori, Vincenzo Ciancaglini, Petar Maksimovic and Hung Nguyen Chan. A backward-compatible protocol for inter-routing over heterogeneous overlay networks, in Proceeding of the 28th Annual ACM Symposium on Applied Computing (SAC), pp. 649-651, 2013.

3. Giang Ngo Hoang, Luigi Liquori and Hung Nguyen Chan. A Scalable Communication Architecture for Advanced Metering Infrastructure. Technical Report.

4. Vincenzo Ciancaglini, Luigi Liquori, Giang Ngo Hoang and Petar Maksimovic. An Extension and Cooperation Mechanism for Heterogeneous Overlay Networks, in Proceeding of the HetNETS Workshop, NETWORKING conference, pp. 10-18, 2012.

5. Vincenzo Ciancaglini, Luigi Liquori, Giang Ngo Hoang. Towards a Common Architecture to Interconnect Heterogeneous Overlay Networks, in Proceeding of the 17th International Conference on Parallel and Distributed Systems (ICPADS), pp. 817-822, 2011.

Chapter 2

Background and State of the art

Contents

In this chapter, we present some basic knowledge on P2P including its definition, the development history, the layer architecture and the applications of P2P. We also provide a literature survey of previous studies on cooperating P2P systems and state the differences between the approach presented in this thesis and the previous ones.

2.1 Background on P2P

2.1.1 P2P definitions

There has no precise definition of P2P in the literature today. Many definitions are proposed trying to capture main features of P2P systems. Following are typical definitions.

G. Camarillo. "A system to be P2P if the elements that form the system share their resources in order to provide the service the system has been designed to provide. The elements in the system both provide services to other elements and request services from other elements" [Camar 2009].

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

R. Schollmeier. "A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P,...) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, ...). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept)" [Schol 2002].

The characteristics of P2P systems according to these definitions include:

Shared resources. Elements or participants in P2P systems share the resources with each other in order to provide the services or content offered by these systems.

Self-organized. P2P systems have ability to self-organize into network topologies.

Dual role. Elements or participants in P2P systems both provide resources to others and request resources from others.

Stability. P2P systems have ability to adapt with failures and accommodate with transient populations of participants or elements.

Autonomy. Participants work without the support of a central servers or authority.

2.1.2 History of P2P

2.1.2.1 The early Internet

ARPANET, the first proto-Internet network, was originally designed in the late 1960s as a peer-to-peer system for the purpose of sharing the computing resources across U.S. The first few hosts, including UCLA, SRI, UCSB, and the University of Utah which were independent computing sites, are connected as equal computing peers. [Oram 2001].

The early stage of Internet was open with no firewall. Two computers can send packets to each other without intermediate intervention. FTP and telnet are then the most popular applications. Although they based on client-server architecture, every host could FTP or Telnet to any other host, i.e. every host act as both client and server.

In this duration, USENET and Domain Name System (DNS), two paradigms of computer networking which include important peer-to-peer components, appeared.

In 1979, USENET was developed to exchange information in the Unix community. Based on Unix-to-Unix-copy protocol, or UUCP, one Unix machine would automatically dial another, exchange files with it, and disconnect. Users could post "news", read messages and exchange messages. USENET introduced the decentralized model of control that in some ways is the grandfather of later peer-to-peer applications such as Gnutella and Freenet [Usenet].

The DNS was originally designed in 1983 to distribute the information of mappings that each of which includes a human-friendly hostname and an IP address, across the Internet. DNS combines the peer-to-peer networking with a hierarchical architecture of information ownership. In DNS system, the namespace is hierarchically organized with each domain has an authority which is the name server keeping mapping hostnames and IP addresses of hosts in that domain. To know the IP address of a host, a peer send the query to the nearest name server, which normally, in turn, delegates the query to a higher authority. Name servers act both as clients and as servers. Several principles in

DNS are used in P2P systems today. For instance, hosts play the dual role of both client and server; the load is naturally distributed over name servers; name servers can query each others, but normally, the communication follows a path up the chain of authority. [DNS, Rfc1034, Rfc1035].

The peer-to-peer communication patterns in early Internet give experiences for designing P2P systems later.

2.1.2.2 The Internet explosion and the appearance of P2P applications

The explosion of the Internet in 1990s involved millions of ordinary users began to use the Internet to exchange email, access web pages, and buy things. This change shifted the equal sharing information model used before to the client-server model.

P2P attracts attention with the popularity of Napster in 1999, a P2P file-sharing application for sharing music. In Napster, central servers indexed the users and their shared files, and returned the information to users who wanted to download the files. A user, upon receive the location information of the sough file, directly exchanged the file with the user who owned that file. One limitation of P2P architecture introduced by Napster is that the central server is the single point of failure. However, the actual problem which took the Napster down in 2001 is the copyright infringement. [Napst]

At the end of Napster, P2P file-sharing networks such as Gnutella [Gnute], Kazaa [Kazaa] appeared removing the single point of failure problem in Napster. These networks introduced decentralized architectures in which peers are equal or a fraction of peers are automatically elected as super peers. These networks allowed users to download various kinds of files such as documents, movies, etc. After Napster was shut down, these P2P networks still exist because there is no single point can take the network down. Many more P2P file-sharing networks have been introduced till now such as Kad [Kad], eDonkey [eDonk] and BitTorrent [BitTo].

Beside file-sharing, P2P technology also has been used in many other areas such as video streaming, instant messaging and distributed computing.

The P2P architecture has been widely used because of its advantages which are as following [Babao 2012].

Scalability. One important property of P2P networks is that the users contribute the resources. When the number of users increases, the capacity of the network also increases because of the additional resources brought by the new users. In contrast, in client-server applications, if the number of users increases, the more resources need to be added for serving the new users. Otherwise, the quality of service can reduce or the service can even stop.

Resilience. P2P architecture eliminates the single point of failure in client-server architecture, thus increase the robustness of the systems.

Cost-efficiency. P2P networks require low cost of installation, management because of following reasons: (i) the users contribute their resources; (ii) P2P networks are self-organized systems; and (iii) they usually, does not require the dedicated servers which are expensive.

However, the P2P networks have limitation of security. The direct exchanging of resources between end users opens the hole to the computers of end users to malicious.

2.1.3 P2P systems

We model the P2P systems in a layer architecture as illustrated in Figure 2.1. A P2P system constitutes three layers: underlying network, overlay network, and application.

- Underlying network is the communication networks to which peers connect for routing packet.
- Overlay network is the network of peers established at the OSI's application level. The peers rely on underlying network for routing packet to each other. Overlay network is responsible for providing P2P services, such as storing and retrieving data, for P2P applications built on top of it. To accomplish its responsibility, overlay network performs many operations such as management of nodes, e.g. constructing, maintaining the topology; management of resources, e.g. locating, replicating resources; management of security, etc.
- P2P application use P2P services provided by overlay network to provide content, services to users. Typical P2P applications include P2P file-sharing applications, P2P instant messaging applications, P2P video streaming applications, etc.

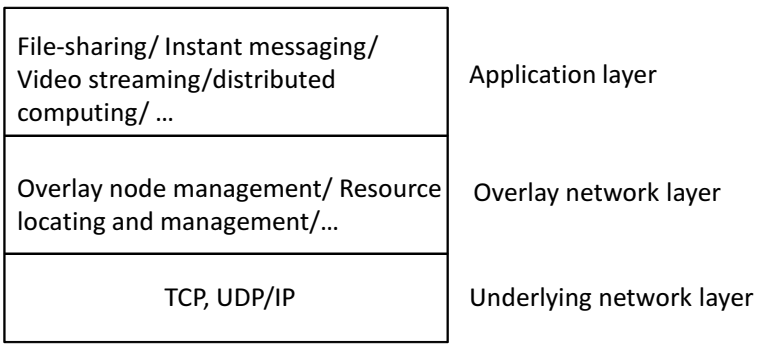

Figure 2.1: Layer architecture of P2P systems

We will investigate the overlay network layer and the application layer of P2P systems in the two follow sections, respectively.

2.1.4 P2P overlays

Many P2P overlay networks have been introduced with different network structures, topologies, routing algorithms, etc. Based on the structure of the overlay networks, these P2P overlays can be classified into structured overlay, unstructured overlay, hybrid overlay and hierarchical overlay.

2.1.4.1 Unstructured overlays

Unstructured P2P systems organizes its peers in a random graph in which either all peers are equal or a faction of peers are elected as super peers having more functions than normal peers. In unstructured overlay, there is no constraint about which node stores a certain content, i.e. content are placed randomly over the nodes in the overlay.

The main operation provided by overlay networks is looking up content. In unstructured overlay, peers use a routing algorithm such as flooding or random walks or hill climbing to query content in the overlay. A peer, who received the query, matches the query locally against its own content.

For purpose of routing, each peer builds and maintains a local routing table containing some neighbor peers. A peer periodically checks the aliveness of these neighbors for removing the unavailable ones and adding the new ones. To ensure the scalability, an overlay limits the maximum number of neighbors that a peer has.

At the bootstrap time, the new peer finds some already existing peers in the overlay via external mechanisms such as via websites. It then contacts with these peers and builds its routing table following the mechanism defined by the overlay.

The unstructured overlay has following advantages:

- It easily supports complex queries, a feature that is expected by many of P2P applications, because all the searches for content are performed locally on each peer.
- It is highly robust in the face of high rates of "churn" - that is, when large numbers of peers are frequently joining and leaving the network [Lv et al 2002, Jin et al 2010].
- It is easy to build and maintain because there is no constrain about its topology [Cherv et al 2008].

Their disadvantages are as following:

- Scalability is a limitation of unstructured overlays due to the high network traffic generated by peers using flooding algorithms which is shown in studies such as [Ripea et al 2002, Marka et al 2002].
- The topology and the location's content are decoupling thus desired data may not be found even if it is exist on the network, i.e. the routing is not exhaustive.

Examples of unstructured overlays are Gnutella[Gnute] , FreeNet [Clark et al 2001], FastTrack/KazaA [Kazaa], Overnet/eDonkey2000 [eDonk] and BitTorrent [BitTo]. We will investigate Gnutella, i.e. an unstructured overlay, in more detail as a demonstration.

Gnutella description. In Gnutella 0.4, i.e. the first version, all peers are equal and peers play the role of both client and server.

Message types. Gnutella employs four different kinds of messages for management and searching.

- PING message is used by peers to discover hosts on network.
- PONG message is used by peer for replying to PING message it received. PONG message contains the information of the peer.
- QUERY message contains search string created by requester.
- QUERY HIT message: a peer, upon receiving the QUERY message, matches again its local shared files for the criteria in the QUERY message. If the match happens, the peer returns the QUERY HIT message containing the necessary information for downloading the file.

Time To Live. Gnutella messages contain a field called Time To Live (TTL) for looping avoidance. TTL is the maximum number of hops that a message can go through. Upon receive a message, each peer decrements the TTL before passing it on to another peer. When the TTL reaches 0, the message is dropped.

Bootstrapping. To join the network, a peer must find at least another node by using address list of possibly available hosts delivered with the software or using updated web caches of known nodes or using UDP host caches. Once the first connection is established, the new peer asks a list of working addresses from its connected host. It then connects to the known hosts to a certain quota and locally caches the untried host for later usage.

Search. To find certain content, a peer sends the query to all of its active connected nodes which then forward the requests, and so on, until reaching the sought content or the TTL reaches 0.

Exchanging files. File downloading is performed directly between two peers, i.e. the peer initiating the search and the peer holding the content.

Maintenance. To deal with the changes in neighbors caused by the join/leave of peers, a peer periodically broadcast PING messages to all of its neighbors, which in turn, forward the message and return the PONG message containing contains information about the peer. A PING message is forwarded until the TTL reaches 0.

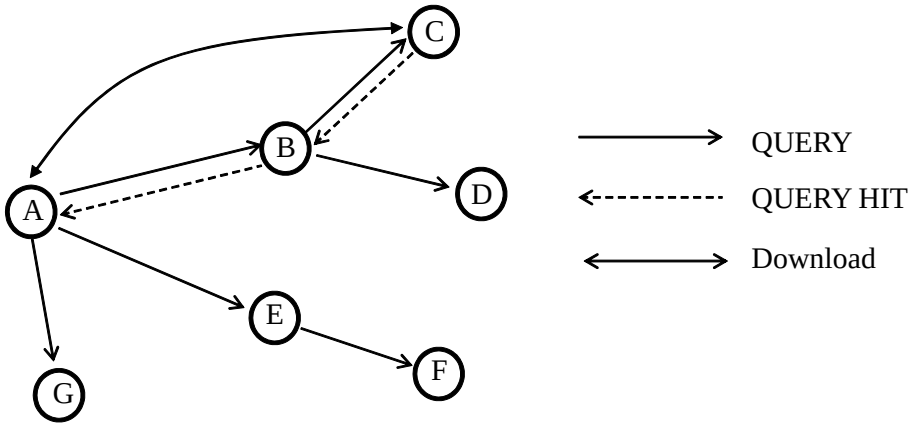

Figure 2.2: An example of search in Gnutella

Example. In Figure 2.2, we illustrate the search using flooding algorithm happen in Gnutella network. In this example, peer A initiates a query for a data which is located on peer C. A sends the query to all of its neighbors, i.e. B, E and G. Upon receiving the query, peer B, in turn, broadcast the query to all of its neighbors, except peer A. Once receiving the query, peer C returns the QUERY HIT response back to A through B. A then directly contact with C for downloading the data.

With the pure decentralized model, Gnutella is very resilient with the churn, i.e. a fraction of peers join/leave the network. However, the flooding search used by Gnutella makes it not scalable.

To deal with the scalability problem of Gnutella 0.4, Gnutella 0.6 introduced a new model with the network consisting of super peers and leaf peers. In this model, a leaf peer connects to a small number of super peers (typically 3) while each super peer connects to more than 32 other super peers. The leaf peers and the super peers periodically exchange the content holding by the leaf peers. The searching happen in the super peer network is done by flooding algorithm. Introduction of super-peers has led to lower network bandwidth and improved scalability [Miloj et al 2002].

2.1.4.2 Structured overlays

In structured overlays, peers are organized into a geometric topology. Data is not placed randomly as in unstructured overlay but at specific locations. This makes the discovery of data more efficient.

Most of structured overlays implement a Distributed Hash Tables (DHT). Chord [Stoic et al 2001], Kademlia [Maymo et al 2002], Pastry [Rowstron et al 2001], CAN [Ratna et al 2001] are examples of DHT. A DHT bases on an abstract key space. Each participating node (peer) or data item is assigned a unique key taken from the key space using consistent hashing. A key is also called identifier. Each DHT defines a distance metric between two identifiers. A value is stored at the node that the distance between the value's key and the node's identifier is minimal.

DHT provides two operations: put(key, value) and get(key) for storing and retrieving of data (or value) on the overlay network. Given a key, DHT route the put or get request to the node which is responsible for the key.

For routing purpose, each node maintains overlay links to a number of other nodes. These links together form an overlay. The information of each of these nodes, including IP address and node's identifier, is stored in the routing table of the local node. The routing forwards the requests across the overlay links closer and closer to the destination node, i.e. in a progressive manner.

At the time of bootstrapping, a peer retrieves information of several peers which exist in the overlay via external mechanisms such as from websites. Upon having the information, the new peer connects with the existing peer and builds its routing table.

To deal with the join/leave of peers, peers in a DHT periodically run a maintenance protocol for updating their routing tables.

There have been many structured overlays are introduced. They can be distinguished by several dimensions such as the maximum number of hops, the geometry topology, the degree of node, overlay maintenance, etc.

- Based on the maximum number of hops a request traverses in an overlay, current DHT can be classified in three categories: one-hop (Kelips [Gupta et al 2002], D1HT [Monnerat et al 2006]), multi-hop (Chord [Stoic et al 2001], CAN [Ratna et al 2001], Koorde [Kaashoek et al 2003]) and variable-hop (Accordion [Li et al 2005], Chameleon [Brown et al 2009]).

- Overlay geometry defines the static model of the graph that routing algorithm constructs in no churn environment. DHT are designed with many overlay geometry, such as Ring (Chord, DSK [Alima et al 2003]), Plaxton Tree (Pastry [Rowstron et al 2001], Tapestry [Zhao et al 2004]), XOR (Kademlia [Maymo et al 2002]), Hypercube (CAN).

- Node degree is the number of entries in the routing table of node. Two important categories are constant degree and logarithmic degree. CAN, Kooder, Hypercup [Schlosser et al 2003], Viceroy [Malkhi et al 2002] are examples of the DHT with constant degree while Chord, Kademlia, Tapestry, P-Grid [Aberer et al 2003] are DHT with logarithmic degree.

Advantages and disadvantages. With the ability to find the unpopular content by guaranteeing a maximum number of messages to locate any peer in the overlays, the advantage of structured overlay is the scalability. However, the structured overlay must maintain lists of neighbors that satisfy specific criteria. This makes them less robust in networks with a high rate of churn (i.e. with large numbers of nodes frequently joining and leaving the network) [Lv et al 2002]. Additionally the structured overlay is unaware of underlying network due to the randomly assignment of key. This means one logical hop can corresponds to multi physical hops, therefore causes the inefficiency in routing.

Kademlia description. Following, as an example of structured overlay, we will describe Kademlia protocol [Maymo et al 2002] in aspects which are general for all structured overlays, including geometry topology, mapping content onto peer, routing table, lookup process, node degree and routing cost, join, leave and maintenance.

Geometry topology. Kademlia organizes its nodes into a binary tree in which node's ids, randomly taken from 160 bit identifier space, are the leaves of tree. The position of a leaf is determined by the shortest unique prefix of the id.

In Figure 2.3, we illustrate the position of a node whose shortest unique prefix is 0100 in a Kademlia binary tree.

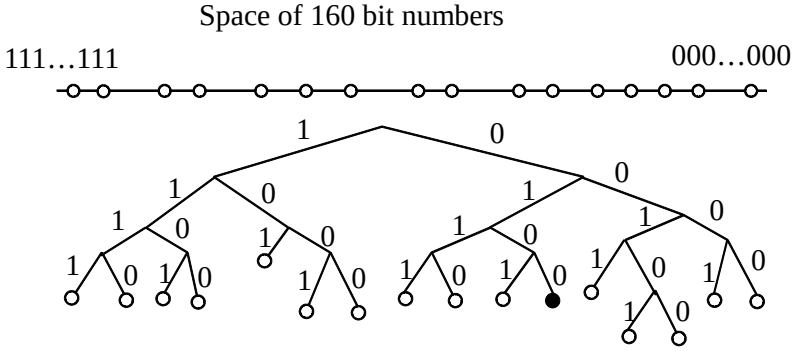

Figure 2.3: A Kademlia binary tree

Distance metric. Kademlia defines its distance metric between two identifiers in the identifier space as the value of the bitwise exclusive or (XOR) of the two identifiers.

Node state. A node divides the binary tree into a series of successively lower subtrees that don't contain the node. The highest subtree contains the half the binary tree that does not contain the node. The next subtree contains the half of the remaining subtree which does not contain the node and so on. The node keeps a list of contact of nodes belonging to each of these subtrees for routing the messages. Contact of a node contains node IP address, UDP port and node ID. This also means that for

each $0 \leq i < 160$, the node keeps list of contacts for nodes of the distance between 2^i to 2^{i+1} from the node.

Mapping items onto node. An item is stored at the node such that the distance between item's id and node's id is minimal.

Lookup process. Kademlia employs iterative and concurrent lookup algorithm. To lookup an ID, the lookup initiator picks α random nodes from the non-empty k-bucket which is closest to the ID or α closest nodes it knows of in case that bucket has fewer α nodes. The initiator then sends in parallel query to these α nodes that each of which possibly returns k closest nodes to the ID it knows of. Upon receiving the results, of the k nodes closest to the ID, the initiator resends the lookup message to α nodes has not yet queried. If a round of sending lookup messages fails to return a node which is closer than the closest already seen, the initiator resends the lookup messages to all of k closest nodes which has not yet queried. The lookup process terminates when k closest nodes it has seen have been queried and have returned the responses. With the prefix matching scheme, Kademlia resolves a lookup in O(log(N)) hops.

Join, leave and maintenance. To join the network, a new node connects to a pre-existing node and then lookups IDs in different subtrees. The learnt nodes in the lookup processes are used to fill its routing table.

To deal with the leave of nodes, Kademlia uses a replacement cache of nodes eligible for replacing failed nodes in the k-bucket. When a node learns a new node and if the perspective k-bucket is full, the new node is inserted into the replacement cache. When a node in a k-bucket is not responsive, it is evicted and replaced by node in the replacement cache which is kept sorted by last time seen with the most recently seen node has highest priority for replacement candidate.

Kademlia uses lookup traffic for maintaining the topology which minimizes the maintenance cost. Because the XOR distance defined by Kademlia is symmetric, a Kademlia node can use new learnt nodes from lookups for updating its routing table.

2.1.4.3 Hybrid P2P systems

Hybrid P2P systems integrate unstructured and structured topologies to exploit the advantages of them. There are many ways of combination. In [Castro at al 2005], graph from structured are used for data placement while search strategies of unstructured overlays are used for locating data. In [Ganes et al 2003], nearby nodes are grouped into small DHTs and the routing between DHTs is performed using unstructured links between them. In [Hui et al 2006], peers in structured overlays are virtually organized in clusters which connected to each other via "long links" to emulate the small world properties [Small-world].

2.1.4.4 Hierarchical overlays

Hierarchical overlays bring the hierarchical structure into the flat structured overlays. A hierarchical overlay organizes its peers into nested overlays, and connects these nested overlays in a tree. To send a message to a peer in a different overlay, a peer sends the

message to the nearest common parent overlay with the destination peer. The hierarchical overlay are designed for improving the performance in such P2P systems that the locality is important in their operations. Many hierarchical overlays has been proposed, including TOPLUS [Garce et al 2003], Hieras [Xu et al 2003], Canon [Ganes at al 2004], Cyclon [Artig at al 2005], etc.

In summary, we present the overlay networks in a classification tree which is adopted from [Buford et al 2010] in the Figure 2.4 below.

2.1.5 P2P applications

2.1.5.1 Content Distribution.

P2P file-sharing. P2P is most successful in file-sharing domain. Typical P2P file-sharing networks include BitTorrent, Kad, Edonkey, Gnutella, etc. In P2P file-sharing networks, peers are organized in structured or unstructured overlays. While the shared files are often kept locally in the owner machines, the information about these files can be distributed to some specific peers or to central server or also kept locally. A peer download certain file in two steps: searching and downloading. In the searching phase, the initiating peer locates the peers who keep the file. It sends the query, often in term of keywords, on the network and the information of matched files is returned to the initiating peer. Normally, after a file is chosen, the peers which keep the file are located and return to the requesting peer. BitTorrent, the most currently used P2P file-sharing system, is one exception. In BitTorrent, user search and download the ".torrent" file, containing the information of tracker which tracking the peers which are sharing the file, from the websites. In the downloading step, the initiator peer directly contacts with the peers who keep the file and then the exchanging of file happen. The downloading operation can use different algorithms such as queuing, swarm, etc.

P2PTV. P2PTV are P2P systems for redistributing video streams in real time. The video streams can be TV channels or can come from other sources. A typical architecture of P2PTV consisting of trackers which keep the information of peers who distribute a certain channel. When a peer wants to view a channel, it connects to the corresponding tracker for the peers are distributing it to receive the video stream from them. In these systems, a user download a video stream and upload it to other users at the same time, thus contributing to the overall bandwidth. Examples of P2PTV application are TVUPlayer [TVUplayer], PPLive [PPLive], Zattoo [Zattoo], LiveStation [Livestation], etc.

Other content distribution systems. P2P technology also is used to build other content distribution applications such as systems for Linux distributions, pre-recorded TV program distribution [BBC-iPlayer], and game updates, etc.

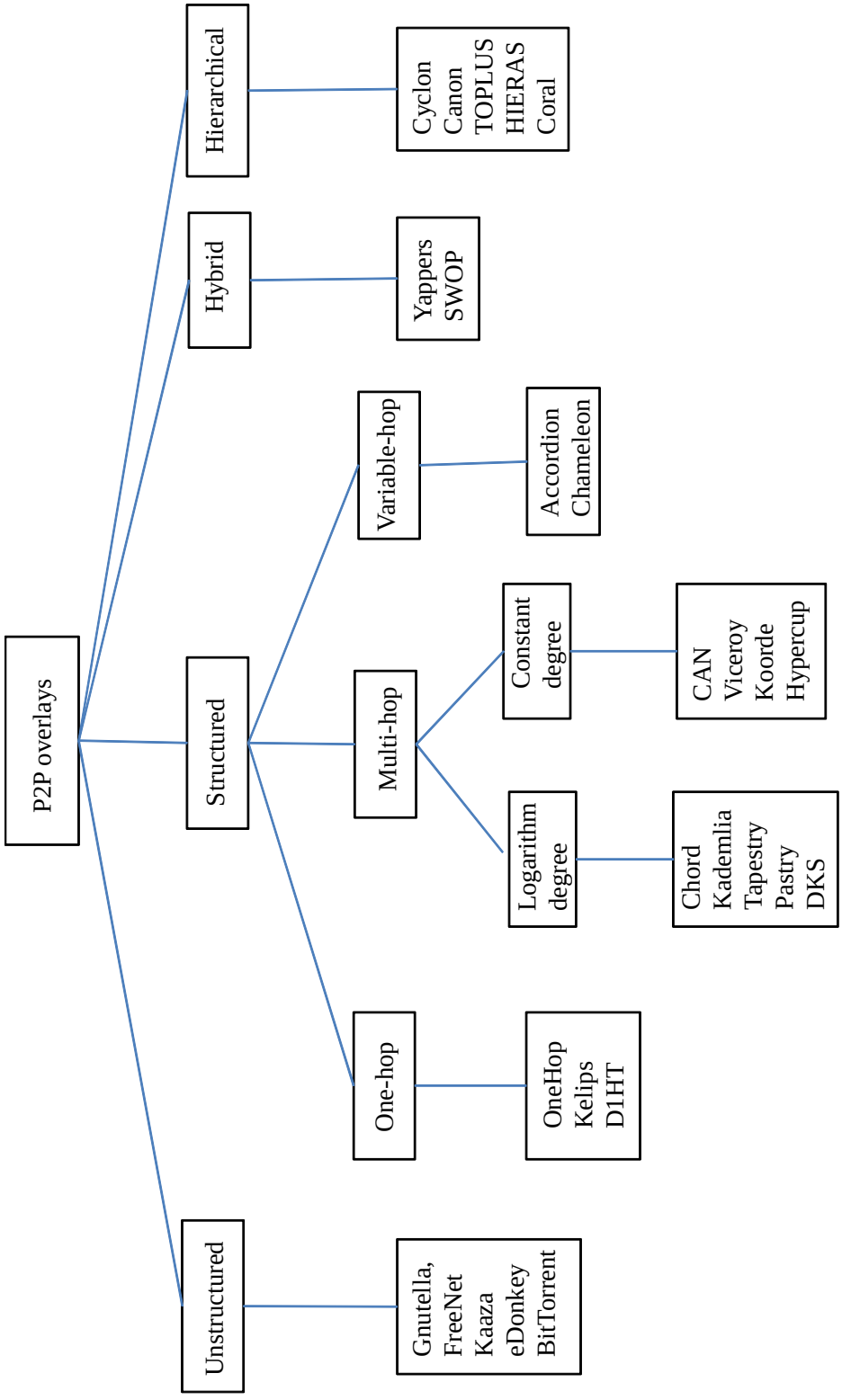

Figure 2.4: Summary of Overlay networks

BitTorrent description. Following we describe the BitTorrent file-sharing system as a typical example of P2P applications for content distribution.

BitTorrent is most widely used P2P file-sharing systems today according to [AppUsage]. As of June 2013, BitTorrent consumed 3.35% of all worldwide bandwidth and more than half of the 6% of total bandwidth consumed by file sharing system.

The architecture of BitTorrent includes a central location for managing users' downloads. To publish a file, an user creates a .torrent file containing information about the file, its length, name, and hashing information, and URL of a tracker the user will register the .torrent file. Once registering the .torrent file with a tracker, the user distributes the .torrent file by conventional means such as web, mail, etc. The user then makes the file available through a BitTorrent node acting as a seed of this file. The file is divided into multiple pieces. This publishing peer is called initial seeder.

For every registered file, the tracker is responsible for keeping track of all the peers who have the file either partially or completely.

To download a file, a peer must first obtain a .torrent file for the file. It then connects to the tracker, specified in the .torrent file, for a list of peers from which it can download pieces of the file. The initiating peer then connects to those peers to obtain the various pieces. If the swarm, i.e. the set of peers are sharing the file, contains only the initial seeder, the initiating peer connects directly to the initial seeder and begins to request pieces. When a peer receives a new piece of the file, it becomes a source of that piece for other peers. If the peer has not had all of the pieces, it is called a leech. Otherwise, it is called a seed.

The Figure 2.5 illustrates the architecture and operation of BitTorrent protocol in which the peer A hosting a file act as the seed uploading the .torrent file to the web server and registering it to the tracker. The peer B then wants to download the file, it announces Get message to the tracker who in turn returns a list of peer involving this file. Upon receiving the list, B contacts and retrieves the pieces from A and C who then also retrieves the file from B.

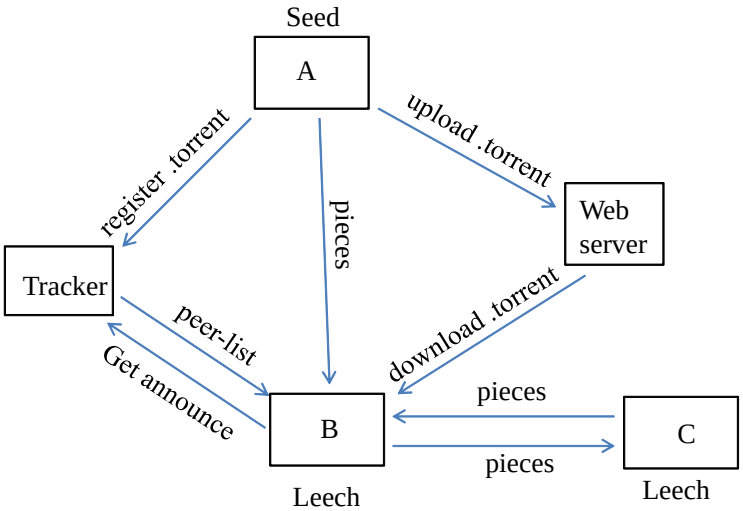

Figure 2.5: BitTorrent architecture and operation

A peer can download pieces of a files in a non-sequentially manner and are rearranged into the correct order once all pieces are downloaded. A peer knows which pieces it needs

as well as which pieces it has and can upload to other peers. This feature help BitTorrent peer resume the download previously halted without the loss of previously information. Therefore, BitTorrent particularly is useful for downloading of large files.

BitTorrent peers employ tit-for-tat scheme [Tit-for-tat] for optimizing their download speed. A BitTorrent peer is prefer to send data to peers that send data back to them. This scheme also allows discouraging free-riders.

On 2005, Azureus 2.3.0.0 [Azureus], a DHT implementation, was released introducing the distributed tracker model in which each peer in the DHT plays the role of the tracker. This model allows the client to use torrents without having a working BitTorrent tracker. Later, many BitTorrent clients such as official BitTorrent client [BitTorrent], μTorrent [μTorrent], BitCommet [Bitcomet], etc support distributed tracker.

BitTorrent originally doesn't support search capability. The Tribler [Tribler] and BitCommet are two BitTorrent clients has incorporated the decentralized search capabilities. With these clients, .torrent files are hosted among peers, instead of on a centralized index sites.

2.1.5.2 Distributed Computing

A P2P distributed computing application breaks a task into multiple independent subtasks and distributes these subtasks to a number of peers in the overlay. These recipient peers perform the tasks using their resources and return the results which will be combined. The advantages of P2P distributed computing is to allow heavy tasks are solved by regular computers instead of powerful servers.

One example of P2P distributed computing system is SETI@home [Seti] which "is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI)". The objectives of SETI@home is to build a huge virtual computer by aggregating the computer processing power offered by voluntary computers which running client software in their idle time. SETI consisting of two parts: the databases server and the clients. The clients search for extra-terrestrial life while the server is responsible for distributing tasks to each participating computer and collecting the results.

2.1.5.3 Communication

P2P technologies are also used in the P2P communication applications including Voice over Internet Protocol (VoIP) applications; Instant Messaging (IM) system; and online chat networks. Examples of P2P communication applications are Skype [Skype], Yahoo! Messenger [Y!M], Pidgin [Pidgin], etc.

Skype, an Internet telephony network, is the most successful system in this category. While Skype protocol is proprietary, a part of the Skype technology based on the Global Index P2P protocol [Baset at al 2006]. Skype distinguishes it from standard VoIP clients by using a P2P model instead of traditional client-server model.

Instant messaging systems allow the instant communication between two or more people over a network using text. IM protocols are various, can be centralized or distributed or mix of them. These systems can, but may not, employ P2P technologies and can provide extra P2P services such as file-sharing, VoIP, etc.

2.1.5.4 Platforms

One application of P2P is to provide the platform for developing the distributed applications using P2P technologies. It provides functions for applications such as discovery, naming, communication, resource aggregation, etc.

One example of P2P platform is JXTA. JXTA aims at building core network computing technology for providing a set of mechanisms supporting P2P computing on any platform, anywhere. JXTA architecture includes three layers: core, service and application. To create a peer-to-peer system, JXTA specifies basic functions which are necessary to support P2P applications and providing them as building blocks for higher-level functions. Core layer supports the creation and deletion of peers, the routing, join, leave of peers, etc. The next layer supports creating peer services such as indexing, searching and file-sharing. The peer applications can be built using provided functions. [RaTiOn 2001] However, Oracle withdrew from the JXTA projects in 2010. [JXTA]

2.1.5.5 Distributed P2P currency

One recent promising application of P2P, which involves many discussion, is Bitcoin, an open source peer-to-peer electronic cash system developed by Satoshi Nakamoto [Bitcoin].

The Bitcoin system is decentralized with no central server or trusted parties. In this system, a user can have some Bitcoin addresses called "wallet" from which they can send or receive bitcoins. Each of the address is a cryptographic public key. The matching private key is stored at certain place such as in digital wallet or mobile device, etc.

The Bitcoin network protocol aims at solving the problems related to creating a decentralized currency and a peer-to-peer payment network. Bitcoins are mathematically generated by a procedure called Bitcoin mining. A coin contains public key of its owner. A user transfers his coin to other user by signing the public key of the receiver to the coin. However, the big problem with this model is double-spending. Bitcoin uses a peer-to-peer network which works as distributed timestamp server to eliminate the double-spending.

2.2 State of the art on cooperation of P2P systems

2.2.1 Classification of cooperation

We classify the studies about cooperation between P2P systems into three following categories: inter-system content sharing, merging systems and inter-system traffic engineering.

- **Inter-system content sharing.** P2P systems co-operate to share the content resources or services with each other. Therefore, users in one P2P systems can exploit resources or services provided by other P2P systems.
- **Merging P2P systems.** P2P systems sharing similarities in goals, the number and the location of nodes can be merged into one system thus eliminating the competition of these systems.
- **Inter-system traffic engineering.** P2P systems co-operate to optimize the routing performance in P2P systems, such as reducing the delay, increasing the bandwidth, etc.

While the objective of inter-system content sharing and merging systems is to enlarge the space of content resources or space of services of users, the inter-system traffic engineering scheme is for improving the performance of P2P systems.

The work done in this thesis belongs to the first category, i.e. allow the sharing content resources and services between P2P systems. Therefore, in this section, we deeply investigate the state of the art on the first categories while briefly survey the works in second and third categories.

2.2.2 Inter-system content sharing

Most of the previous works in this category, except [Konis at al 2006], only addressed the problem of routing a request between overlays which we hereby call inter-overlay routing. In [Konis at al 2006], they proposed a scheme not only for inter-overlay routing but also for exchanging data in pure flooding P2P file-sharing networks.

Inter-overlay routing. Efficient inter-routing between overlays has served as an inspiration to a number of research efforts. These researches follow two approaches: inter-routing via logical links [Konis at al 2006] and inter-routing via co-located nodes [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012].

In [Cheng 2007], the authors proposed a solution for inter-routing between distributed hash tables (DHT). One subset of peers in a DHT is co-located nodes, while another subset of peers in that DHT is chosen as trackers to keep the information of those co-located nodes. The queries from peers in one DHT are sent to the trackers that, in turn, forward them to co-located nodes to reach other DHTs. The mechanisms of selecting trackers require the modification of all peers. Also, no simulations or experiments have been made.

In [Liquo at al 2009], the author introduced a protocol named Babelchord for inter-routing between Chord overlays. A subset of peers in each overlay runs the Babelchord protocol, serving as gateways that forward messages between these overlays if the messages randomly touch Babelchord nodes in their paths. All peers are modified to add some extra information to their messages.

In [Liquo at al 2010], the authors introduced a protocol named Synapse for inter-routing between different DHTs. They presented two models: *white box* and *black box*. In a nutshell black box ensures backward-compatibility while white box is not. The white box model is an extension of Babelchord [Liquo at al 2009], allowing inter-routing between overlays running different DHTs instead of only between overlays running Chord. The black box model does not require the modification of all peers. It uses a set of additional overlays, called Control Overlays, to cache the information about performed lookups, instead, such as the mapping (hashed key, key), lookup results. In this model, if a query originating from one overlay randomly touches a Synapse node, it will be forwarded to other overlays if the information, needed for forwarding the request, has been cached in the Control Overlays in the previous lookups. However, in this paper, the black box model was not evaluated.

In [Cianc et al 2011, Cianc at al 2012], we introduced an improved version of Synapse protocol in which a new black box model was proposed. Some mechanisms for Synapse peers to detect each other are introduced, while the Control Overlays concept of [Liquo at al 2010] was dropped. In a nutshell, upon receiving a query, a

new Synapse node can forward that query to other new Synapse nodes in addition to launching the query on its connected overlays. The black box model in [Cianc et al 2011, Cianc at al 2012] was evaluated via experiment on two kind of DHTs namely Chord and Kademlia.

Content cooperation of P2P file-sharing systems. In [Konis at al 2006], the authors proposed mechanisms for cooperating purely query flooding-based file-sharing networks. In their model, several pairs of peers from two networks establish logical links between the two networks serving as bridges to transfer the search requests and the found files between them. However, this work is only for unstructured, query flooding-based file-sharing networks which constitute a small portion of current file-sharing networks.

Discussion. Each P2P system is usually built with its own well-defined semantic. Therefore, one of the fundamental tasks is the transcoding of the messages and data between different P2P systems.

In all previous works, the function of transcoding requests between P2P systems is "built in" in the routing function, i.e. the transcoding is defined by routing protocols. Moreover, the works [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012] only deal with the transcoding between hashed keys in different DHTs. Additionally, they do not mention how are the data or messages which are not queries transferred/transcoding between overlays. The work [Konis at al 2006] only deals with the cooperation of two unstructured file-sharing networks thus the transcoding of queries is simple and they did not mention how their transcoding is.

Due to the transcoding function is built in the routing function while the transcoding schemes are not complete and so simple, all of the previous solutions are not practical [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012], i.e. it is difficult to cooperate P2P systems or limit in application area, i.e. [Konis at al 2006] is only for unstructured and purely flooding based queries.

All of previous solutions are only applicable homogeneous overlays. The solutions in [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012] are only for inter-routing between DHTs, i.e. structured overlays. The model in [Konis at al 2006] is only for inter-routing between flooding-based unstructured overlays.

The solution in [Liquo at al 2009, Liquo at al 2010] requires the modification of all peers in overlays, i.e. peers which are unaware of new protocols cannot operate, which is not practical in reality. Solutions in [Konis at al 2006, Cheng 2007, Cianc et al 2011, Cianc at al 2012] ensure the backward compatibility in the sense that the native peers which are not aware of new protocols can operate normally, which is suitable for inter-routing between existing P2P overlays.

Only [Konis at al 2006] dealt with the cooperation of applications running on top of overlays. However, in their model, applications are only the unstructured, flooding-based P2P file-sharing applications with the same mechanisms of search and download file which is quite simple to solve.

The authors in [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012] did not deeply investigate how the applications can exploit their models. The P2P applications are normally not just send a request to other peers and receive

the results from them. Instead, they have more operations which require the communication and exchanging of data between the requesting peer and the peers which provides content resources or services.

Distinguishing the current work from others. Here we introduce the fundamental difference between current work and previous works. Other differences are presented in Chapter 3 and Chapter 4 which describe the current work.

In the model proposed by current work, the inter-overlay routing function and application bridging function are decoupled instead of coupling as in previous works. The model consists of two layers: the inter-overlay routing layer and the cooperation application layer. The inter-overlay routing layer only provides the routing service for the application on top of it. In the application layer, the cooperation application maps the interface of the original application into a common interface to address the incompatibilities between various original applications. By leaving the cooperation application to defined the transcoding schemes, the architecture introduced by current work achieve more flexible comparing to the previous ones. This is the fundamental difference between current work and previous works.

Other important difference between current work and previous ones is the current work is applicable to heterogeneous overlays, i.e. both structured overlays and unstructured overlay while all of previous works are only applicable to homogeneous overlays, i.e. either structured overlays or unstructured overlay.

2.2.3 Merging overlays

Certain works have focused on merging several overlays into one. In [Datta at al 2006], the authors provide an analysis of the problem of merging two different overlays, while in [Shafa at al 2009], the authors give an algorithm describing the merging mechanism of two ring-based DHTs which requires modifications of the key space, as well as the rearrangement of keys and requires to modified all of the peers in two DHTs.

Discussion. This approach eliminates the competition of P2P systems and the need of inter-systems co-ordination. However, merging systems require the changes from routing to data presentation in all peers, task which are fairly expensive in terms of time and power processing. Additionally, require to modify all peers is impractical for existing overlays. On the other side, the merging of heterogeneous overlays still remains impossible.

2.2.4 Inter-system traffic engineering

In [Kwon at al 2005] author proposed an overlay internetworking architecture constitutes home overlay networks and super overlay called Synergy network including a number of peers from these normal overlays. A route on home overlays can be replaced by a route in the Synergy overlay, i.e. cross multiple home overlays, if the route in Synergy overlay brings better performance.

Authors in [Jiang at al 2005] investigate the interaction between multiple overlays, which are constructed on top of a physical network layer. They proposed the overlay optimal routing policy and model the interaction between multiple overlays using this routing policy in a game theoretic approach.

In [Liao et al 2006], authors introduced an inter-overlay optimization based scheme in which the resources are shared among different overlays to optimize streaming paths, improve service delay and streaming reliability, balance the traffic, etc.

Chapter 3

A Backward Compatible Framework for Cooperation of Heterogenous P2P Systems

Contents

3.1 Introduction

In this chapter, we introduce a new model for sharing content between heterogeneous P2P systems. This model consists of two parts, which bridge the involved systems at the routing and application layers, respectively.

The first part is the OGP routing framework, which is composed by two protocols namely Overlay Gateway Protocol (OGP) and lightweight OGP. The OGP protocol, an extension of Kademlia, allows efficient routing among existing heterogeneous overlay networks. OGP is run only by a small number of peers from each of the standard overlays, in addition to their native protocols. These peers form a super-overlay (the OGP overlay) equipped with efficient algorithms to perform unicast, broadcast, and multicast of messages from one standard overlay to the others. Peers forming the OGP overlay act as gateways for peers especially created for taking advantage of OGP which run the

lightweight OGP protocol, and can reach across standard overlays they are not members of. The lightweight OGP protocol is run by lightweight peers, i.e. peers with even small power and bandwidth capacities, to take advantage of the OGP overlay to reach the standard overlays of which they are not members. The OGP and lightweight OGP protocols together form a framework for inter-routing between heterogeneous standard overlays.

The second part of the model is a cooperation application that makes use of the OGP routing framework, and which is responsible for bridging the P2P systems at the application layer with tasks such as transcoding the messages and data from formats of the P2P systems to intermediary formats and vice versa. Since the particular tasks of the cooperation application depends on the application domain, in this chapter we only describe the principles of the cooperation application.

Our original approach ensures *backward-compatibility*, in the sense that:

- Native peers can continue to operate normally without any modifications, in the presence of new protocols.
- Peers that are aware of new protocols can achieve resources from different systems in a transparent way.

The contribution of this chapter is the introduction of a new model for cooperation between heterogeneous P2P systems consisting of a new framework for efficient inter-routing between heterogeneous overlays and principles underpinning the cooperation application for bridging these P2P systems at the application layer.

The rest of this chapter is structured as follows: Section 3.2 presents the system model, which was the motivation of this chapter. The routing framework based on OGP and lightweight OGP protocols and the cooperation application are described in Section 3.3. Section 3.4 evaluates the model. Finally, in Section 3.5, we present our conclusions and outline future work.

3.2 Framework Overview

3.2.1 Classification of peers

In our framework, the peers are classified into three following categories:

Full OGP peers, hereafter denoted as FOGP peers, simultaneously belong to one P2P system and the OGP overlay. In addition to their native protocols, they also run the OGP protocol and the cooperation protocol. They route messages from one P2P system to the others via the OGP overlay and serve as gateways for lightweight OGP peers to reach P2P systems to which they do not belong.

Lightweight OGP peers, denoted as LOGP peers, take advantage of the inter-overlay routing provided by the OGP overlay. They belong only to one P2P system, do not participate in the OGP overlay, but keep a list of FOGP peers. In addition to their native protocols, they run the lightweight OGP protocol for communicating with FOGP peers and the cooperation protocol. LOGP peers are introduced: (*i*) to reach P2P systems they are not members of with low cost in terms of power processing and bandwidth, and (*ii*) to improve the scalability of the cooperation system by reducing the number of FOGP peers and the size of OGP overlay.

Blind peers are peers that belong to only one P2P system, are not aware of the existence of the new protocols and use only their native protocols.

3.2.2 Inter-routing schemes

The inter-routing algorithms are the heart of OGP protocol. The OGP protocol provides three new and efficient schemes that enable inter-routing between standard overlays: *OGP unicast*, *OGP multicast* and *OGP broadcast*. A FOGP peer can use any of these schemes. For the sake of brevity, the operation of routing a request to a random FOGP peer belonging to the destination standard overlay is hereby described as routing that request to the destination overlay.

OGP Unicast. In the OGP unicast scheme, FOGP peers route requests into only one destination overlay different from the one the request originated from.

OGP Multicast. With the multicast scheme, a FOGP peer can selectively choose multiple destination overlays, and all of the responses are returned to the original sender.

OGP Broadcast. With the broadcast strategy, all standard overlays are chosen as destination, and all of the responses are returned to the sender, just like with the multicast.

3.2.3 Structure of a FOGP peer and a LOGP peer

A FOGP peer, see Figure 3.1, has several components:

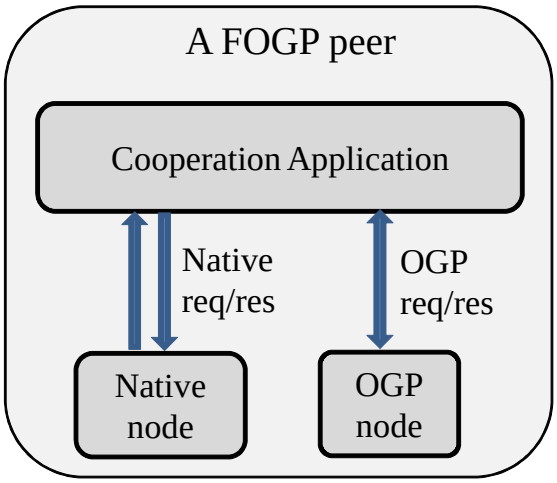

Figure 3.1: A FOGP peer

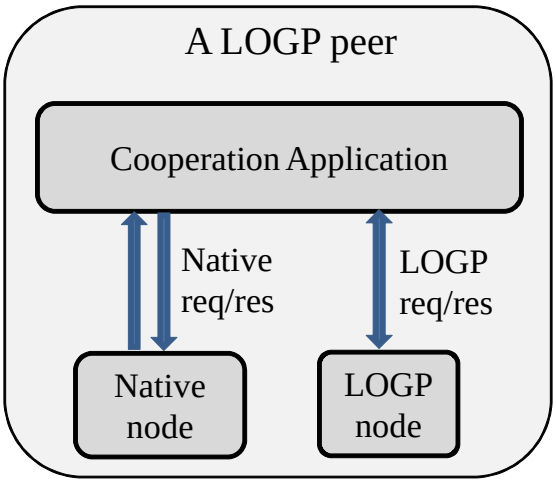

Figure 3.2: A FOGP peer

Native node. The Native node participates in the P2P system to which the FOGP peer belongs, launches requests on this P2P system and returns the results to the cooperation application.

OGP node. The OGP node participates in the OGP overlay and provides unicast, multicast and broadcast inter-routing for the cooperation application.

Cooperation application. The Cooperation application can launch the request on a P2P system via the Native node, on the OGP overlay via the OGP node, or perform certain tasks specific to application domain, and receive the results. It performs the transcoding of messages and data at interface level between formats of P2P systems and intermediary formats.

The structure of a LOGP peer, see Figure 3.2, is the same as the structure of a FOGP peer except that the OGP node which runs OGP protocol is replaced by the LOGP node which runs LOGP protocol.

3.2.4 Cooperation examples

In Figure 3.3, two scenarios are shown to illustrate the cooperation of three P2P systems in which a FOGP peer and a LOGP peer lookup information at overlays they are not members of. The three smaller ovals, denoted by `O1`, `O2` and `O3`, represent standard overlays the P2P systems based on, while the largest oval represents the OGP overlay. The black squares `A`, `B`, `D`, and `G` represent FOGP peers, the black circles `F`, and `C` represent LOGP peers, while the white circles `E`, and `H` represent blind peers. Solid lines represent requests, while dashed lines represent responses.

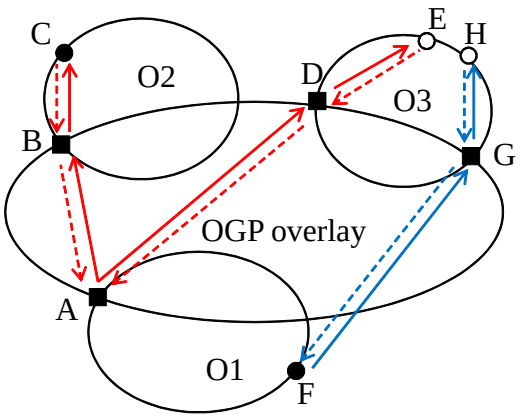

Figure 3.3: Examples of cooperation

First scenario. The FOGP peer `A` is looking for some information which is located at the LOGP peer `C` in overlay `O2` and the blind peers `E` in the overlay `O3`. `A` send the request to the FOGP peer `B` and the FOGP peer `D`, belonging to `O2` and `O3` respectively, via its OGP node, using OGP broadcast routing. Upon receiving the request, `B` and `D` reconstruct the request to be in accordance with the possibly different format defined by the native protocols of `O2` and `O3` respectively, then forward it to `C` and `E` via their Native nodes. `C` and `E` then send the responses back to `B` and `D`, which reconstruct the responses to follow the format defined by cooperation protocol, and send it, along with their contact information for later communication, back to `A`, via their OGP nodes.

Second scenario. The LOGP peer `F`, belonging to overlay `O1`, is looking for some information located at the blind peer `H` in overlay `O3` . It forwards, via OGP node, using OGP unicast routing, the message to `G` which is a FOGP peer in overlay `O3`. Upon

receiving the message, G converts the message in accordance with the native protocol of O3, and forwards it to H via its Native node. The return path takes us back through G to F, following the native protocol of O3 first, and then the OGP protocol.

3.3 Framework description

3.3.1 OGP protocol

3.3.1.1 ID assignment

OGP identifies each standard overlay by a unique n-bit number we denote by netID. A FOGP peer is assigned an unique $(n + m)$-bit identifier, denoted by ID, consisting of two parts: the n-bit identifier of the standard overlay to which that peer belongs (netID), and a random m-bit number denoted by nodeID. Given this, and using | as a concatenation operator, we have that:

$$\texttt{ID} = \texttt{netID} \mid \texttt{nodeId}$$

3.3.1.2 The distance metric

A FOGP peer calculates the XOR distances, which is defined in Kademlia protocol, from itself to other FOGP peers and uses these distances to internally represent these nodes as a binary tree with the leaves of the tree are the shortest unique prefix of these distances. One important property of this binary tree is that all FOGP peers connected to the same standard overlay share a single subtree. Let the identifier of the current node be netIDi | nodeID. By properties of the XOR distances, we can easily see that the distance between the current node and any of the peers connected to the same overlay will share the same n-bit prefix, and, therefore, the same subtree.

In Figure 3.4, we illustrate the binary tree representing FOGP peers from the view of the FOGP peer whose distance metric is 000...000. In this case, we have that $n = 2$, i.e. netID is represented by 2 bits. Here, FOGP peers can belong to one of the four standard overlays, whose identifiers are netID1, netID2, netID3, and netID4.

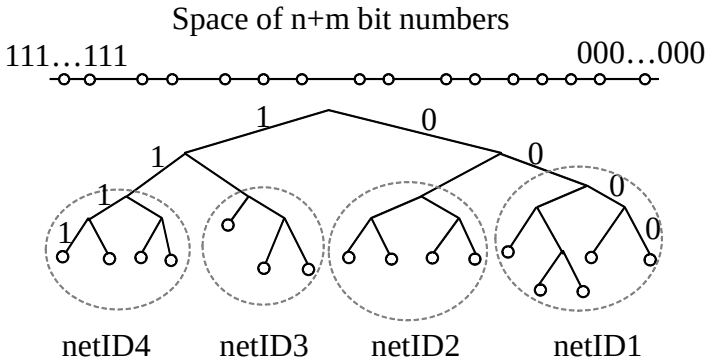

Figure 3.4: A Binary tree for OGP peers with n=2

3.3.1.3 The node state

A node of many popular DHTs such as Chord, Kademlia selects typically O(logn) routing entries to peers with exponentially increasing distance in the id space with n is the number of nodes in the system. This principle allows the DHT to achieve its scalability and guarantees an expected routing cost of O(logn) between any two peers in the network. In OGP, by adding more pointers to some distance spaces farthest from the node identifier, we reserve the scalability of typical DHTs and ensure the efficiency of the OGP routing algorithms in terms of routing cost as well as make the broadcast and multicast algorithms robust to churn and make the load on nodes in one round of broadcast algorithm close to balance.

Let p be the number of bits of identifiers of FOGP peers, i.e. $p = m+n$. Additionally, let U and V be two non-negative integer parameters which tune the ID space division and the number of hops a message traverses. Let U and V satisfy (i) $U \geq 1$, and (ii) $U \cdot V \leq p$, and will be used throughout Section IV.

In OGP, the entire distance space is divided using the following rules:

1. The entire distance space is divided into $p + 1$ spaces, numbered from 0 to p, which, respectively, cover distances from 0 to $2^0 - 1$, and from 2^{i-1} to $2^i - 1$, with $1 \leq i \leq p$, i.e. similar to the rules of typical DHTs.
2. The $U \cdot (V - 1)$ farthest spaces, i.e. the spaces numbered from p down to

   $$p - U \cdot (V - 1) + 1$$

   are further divided into 2^{U-1} equal subspaces; The U farthest spaces of the remaining spaces, i.e. the spaces numbered from

   $$p - U \cdot (V - 1)$$

   down to

   $$p - U \cdot V + 1$$

   are, respectively, further divided into 2^{U-i} equal subspaces, with $1 \leq i \leq U$.
3. The remaining spaces, i.e. the spaces numbered from $p - U \cdot V$ down to 0, are not divided any further.

The deepest subspaces, i.e. those are not further divided, obtained by applying Rule 2 and Rule 3 are the final spaces. To route messages, a FOGP node keeps in its routing table a group of pointers up to k to each final space with k is a system parameter. The group of pointers is called k-bucket as in Kademlia paper [Maymo et al 2002] for simplicity. Each pointer has information about a node belonging to the final space.

Example. Consider an 8-bits identifier overlay in which 7 first bits of identifiers are used for representing the `netID` part. Assume that the overlay is fully occupied by peers and let $U = 3$ and $V = 2$. Consider the FOGP peer, denoted by `A`, with the `ID` of 00000000. The space division from the view of this peer is illustrated in Figure 3.5. Final spaces are represented by black circles. `A` divides the entire space into 9 subspaces, numbered from 0 to 8. Each of the spaces numbered from 8 down to 6 are further divided into $2^{U-1} = 4$ equal subspaces. Each of the subspaces numbered from 5 down to 3 are, respectively, further divided into $2^{U-i} = 2^{3-i}$ equal subspaces with $1 \leq i \leq 3$, i.e. into 2^2, 2^1 and 2^0 equal subspaces, respectively.

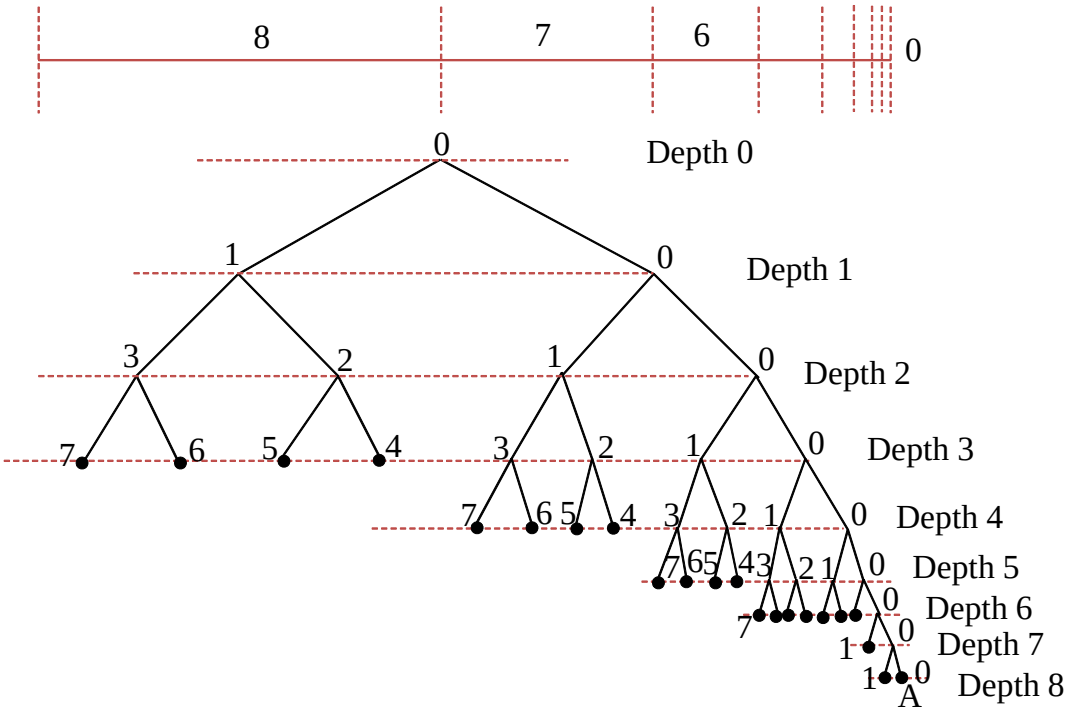

Figure 3.5: An example of space devision

3.3.1.4 The routing table

We adopt the principles of Kademlia in building the routing table of FOGP peers. We divide the routing table in buckets. Each bucket contains nodes with some common prefix in their `IDs`. The routing table of FOGP node can be seen as a binary tree in which each leaf of the tree is a bucket. The position of a bucket in the binary tree is defined by the shortest unique prefix of that bucket.

Like Kademlia, nodes in the routing tree of OGP nodes are allocated dynamically. Initially, the routing tree of a node u has one bucket covers the whole identifier space. Upon learning a new node, the node u inserts the new node into the appropriate bucket. If the bucket is not full, the new node is simply inserted. Otherwise, if the bucket is full, and the bucket is not represent a final space, the bucket is split into two new buckets represent two equal subspaces split from the space represented by the old bucket.

The process of building the routing table of the node with the distance of 00..00 is illustrated in the Figure 3.6. This example, parameters U and V are set as following: U = 2 and V = 1.

In the evolution of the routing table of a node, the whole space is split into subspaces that each of which is further split. The node keeps information of all of those spaces in its routing table. The Figure 3.7 illustrates the spaces in the routing table of the node with the distance of 00..00. In this figure, the whole space S_0 is divided into two equal subspaces S_1 and S_2. The subspaces S_1 and S_2 are further divided and so on. All of those spaces form a tree of spaces.

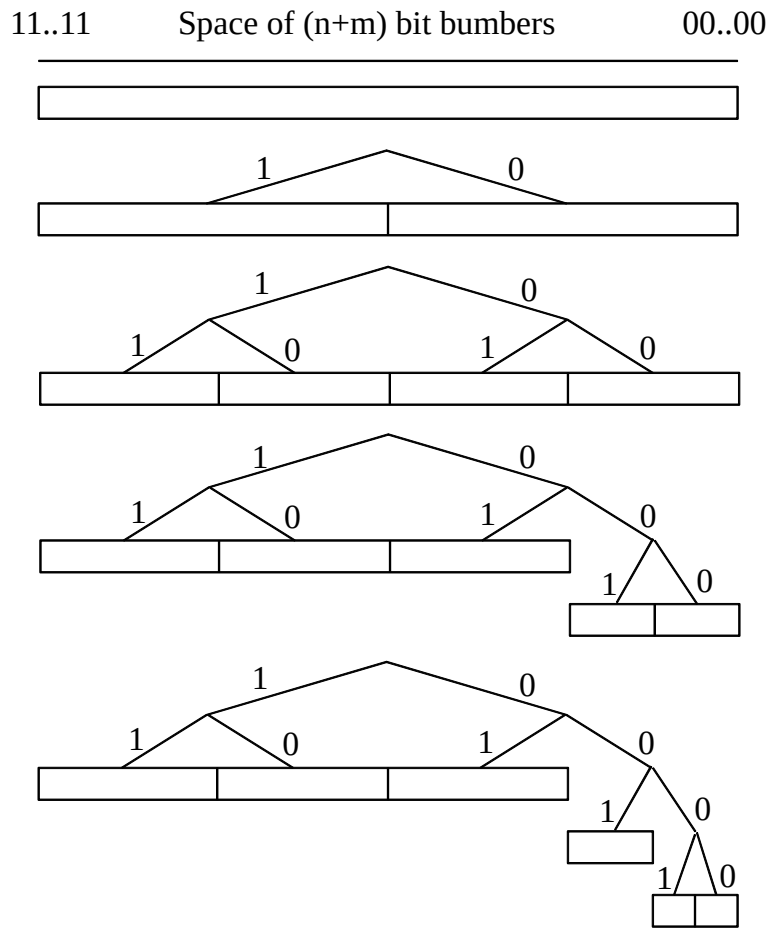

Figure 3.6: The evolution of the routing table

FOGP peer list. An FOGP peer keeps a fix-sized list of other FOGP nodes, belonging to the same n-level subtree with it, which is provided to applications to exploit. Upon learning a new node, if the new node belongs to the same n-level subtree with current FOGP node, the current node inserts the new node into its FOGP peer list if this list has fewer contacts than the limit size. If the FOGP peer list is full; the new node is inserted to the candidate list. A FOGP node periodically runs the check aliveness process to remove the dead nodes from the FOGP peer list and the perspective candidate list. A dead node in the FOGP peer list is replaced by the first alive node in the candidate list.

3.3.1.5 Routing in OGP

Definitions. We first introduce several notions and definitions.

***n*-level subtree.** A *n-level subtree* of the OGP binary tree (illustrated in Figure 3.4) is a subtree whose prefix length equals n. This means that all nodes connected to one standard overlay belong to a n-level subtree.

By "sending a message to a subtree" we mean "sending a message to a random node belonging to the said subtree". We will henceforth use this expression, for the purpose of brevity.

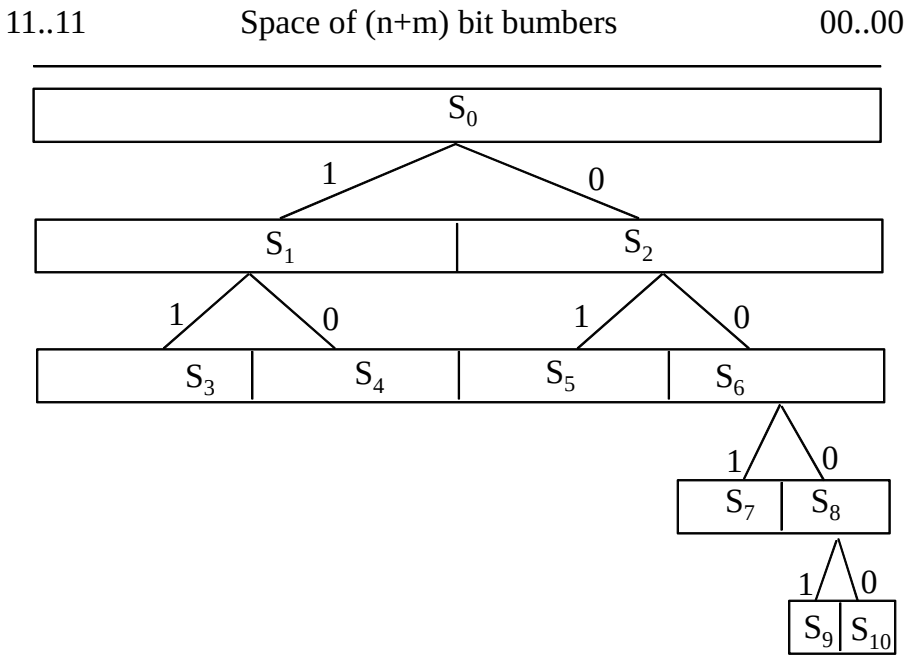

Figure 3.7: The tree of space

Routing schemes. OGP provides three kinds of routing, namely: n-level unicast, n-level multicast, and n-level broadcast.

- The n-level unicast is an algorithm used by a FOGP peer to send a message to an n-level subtree of which it is not a member.
- The n-level multicast is used to send a message to a group of n-level subtrees of which it is not a member.
- The n-level broadcast is the sending of a message to all of the n-level subtrees of which it is not a member.

In all of the cases, each n-level subtree only receives one message.

***Range* of a distance space.** The *Range* of a distance space `S`, denoted by Δ_S, is the `XOR` between the maximal (`UB`) and minimal (`LB`) numbers in `S`:

$$\Delta_S = \texttt{UB} \oplus \texttt{LB}.$$

***Depth* of a distance space.** The *Depth* of a distance space `S` in the space tree of a FOGP peer, denoted by $\delta_{\texttt{S}}$, is the number of bits of the space's prefix in that tree. Similarly to "sending a message to a subtree", we hereby write "sending a message to a subspace" instead of "sending a message to a random node belonging to the said subspace", for the sake of brevity.

The routing schemes provided by the OGP protocol are presented in the following sections.

3.3.1.6 n-level unicast.

Algorithm. The n-level unicast is a greedy algorithm aimed at sending a message to an n-level subtree, knowing its n-bit prefix $\mathtt{P_n}$. The sending node, which is a FOGP node and has the identifier $\mathtt{ID_0}$, first generates an m-bit random number $\mathtt{R_m}$ and concatenates it to $\mathtt{P_n}$ to form an $(n+m)$-bit identifier:

$$\mathtt{ID} = \mathtt{P_n} \cdot 2^{\mathtt{n}} + \mathtt{R_m}.$$

The initiator node then sends a $\mathtt{REPLICATE}(\mathtt{message}, \mathtt{ID})$ request to the FOGP node in its routing table closest to $\mathtt{ID}' = \mathtt{ID} \oplus \mathtt{ID_0}$. Upon receiving the $\mathtt{REPLICATE}$ request, the recipient node checks if its identifier $\mathtt{ID_i}$ and $\mathtt{ID}$ have the same n-bit prefix. If so, the unicast is completed. Otherwise, the recipient node forwards the $\mathtt{REPLICATE}$ request to the FOGP node in its routing table closest to $\mathtt{ID}' = \mathtt{ID} \oplus \mathtt{ID_i}$. If there is no node in its routing table closer to $\mathtt{ID}'$ than itself, the recipient node drops the request.

Example. An example of n-level unicast algorithm is illustrated in the Figure 3.8. In this example, $n = 2$ and four overlays are indentified by $netID1$, $netID2$, $netID3$ and $netID4$.

Node A belonging to the overlay $netID1$ wants to lookup certain content in overlay $netID3$. It first generates a random m-bit number $\mathtt{R_m}$ and concatenates it with $netID3$ to form $(n+m)$-bit identifier:

$$\mathtt{ID} = \mathtt{netID3} \cdot 2^2 + \mathtt{R_m}$$

Node A then sends the request to the node closest to ID in its routing table which is node B belonging to $netID4$. Upon receive the request, node B check to see if the ID belongs its n-level subtree by comparing the n first bits of its identifier and the ID. Because the n first bits of the two identifiers are different, node B forwards the request to the node closest to ID in its routing table which is node C. Upon receive the request, node C knows the ID belongs to its n-level subtree thus it will proceed the request.

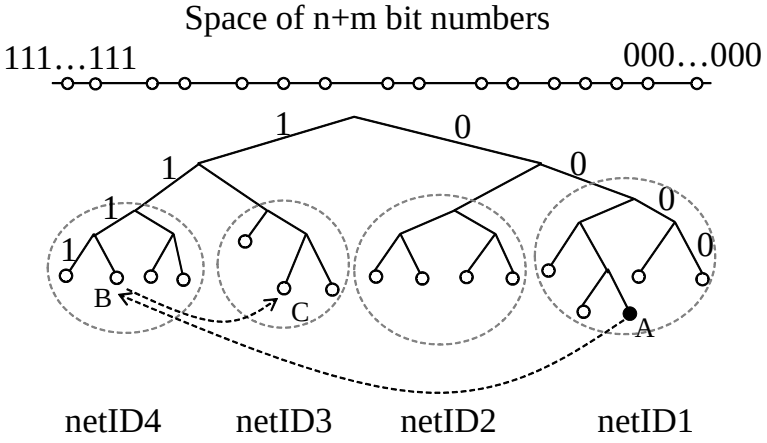

Figure 3.8: Example of n-level unicast

Discussion and analysis. By n-level algorithm, the message jumps from one n-level subtree to an other n-level subtree to approach closer and closer to the destination n-level subtree. At each n-level subtree, the request touches only one node. Hence, we can assume that each n-level subtree is a virtual node in the overlay with n-bit identifier space.

Because the distance space division in the routing table, the distance of the message from the destination n-level subtree is reduced at least 2^U per round of request sending in the V first rounds and is reduced at least twice per round of request sending in the round after. Assume that the number of n-level subtrees in OGP overlay is K. If $log_{2^U} K < V$, after $log_{2^U} K$ rounds of sending the request, i.e. message traverses through $log_{2^U} K$ hops, the distance from the message to the destination subtree is:

$$\frac{2^n}{2^{U \cdot log_{2^U} K}} = \frac{2^n}{K}$$

Because the n-bit prefixes of n-level subtrees are random numbers, the number of the n-level subtrees belonging to the distance $\frac{2^n}{K}$ from the destination n-level subtree is 1, with high probability. That n-level subtree is the destination n-level subtree itself. Thus, if $log_{2^U} K < V$, it takes $O(log_{2^U} K)$ hops to reach the destination. Otherwise, if $log_{2^U} K > V$, then after V hops, the distance from the message to destination subtree is:

$$\frac{2^n}{2^{U \cdot V}}$$

The number of n-level subtrees that stay in this distance is:

$$\frac{\frac{2^n}{2^{U \cdot V}}}{\frac{2^n}{K}} = \frac{K}{U \cdot V}$$

With the same analysis, after

$$log_2 \frac{K}{U \cdot V}$$

hops more, the message touches the destination subtree with high probability. Therefore, if $log_{2^U} K > V$ the message arrives the destination n-level subtree after

$$V + O(log_2 \frac{K}{U \cdot V})$$

hops.

3.3.1.7 n-level broadcast.

Algorithm. This mechanism is used by a FOGP node to send a message to all n-level subtrees to which it does not belong.

The main idea is that the initiator node sends 2^U messages to 2^U nodes belonging to 2^U equal subspaces in its routing table which, together, cover the entire distance space. Each subspace contains at least one n-level subtree. The node belonging to a subspace and receiving the message is responsible for the further broadcast of the message in that subspace, by repeating the sending operation of the initiator node, except that the entire distance space is replaced by the subspace it is responsible for. This process is repeated at most V times. After that, upon receiving the message, a FOGP node sends the message

to all final spaces in its routing table containing at least one n-level subtree, which belong to its responsible space, asking one FOGP node per final space to be responsible for that space. In all cases, a recipient node always excludes the subspace covers the n-level subtree containing it which already received the message from its responsible space before continuing to send the message. A node stops sending messages if the space it is responsible for has only one n-level subtree. The entire process stops when all of the n-level subtrees have received the message.

The n-level broadcast algorithm can be sketched as follows:

The initiator node. The initiator node sends the $\texttt{REPLICATE}(\texttt{message}, \Delta_{\texttt{S}_\texttt{i}}, \Delta_\texttt{i})$ request to every subspace $\texttt{S}_\texttt{i}$ which satisfy the following conditions:

1. $\delta_{\texttt{S}_\texttt{i}} \leq \min(\texttt{n}, \texttt{U})$;
2. $\texttt{S}_\texttt{i}$ does not have any children $\texttt{S}_\texttt{j}$ which satisfy the previous condition;

with $\delta_{\texttt{S}_\texttt{i}}$ is the depth of the subspace $\texttt{S}_\texttt{i}$ and $\Delta_{\texttt{S}_\texttt{i}}$ is the range of $\texttt{S}_\texttt{i}$. Assume that Δ is the range of one n-level subtree. $\Delta_\texttt{i} = \Delta$ if $\texttt{S}_\texttt{i}$ contains the initiator node. Otherwise, we have $\Delta_\texttt{i} = 0$. For the subspace $\texttt{S}_\texttt{i}$ contains the node, the message is sent to the n-level subtree which is closest to the initiator.

The condition $\texttt{Depth}(\texttt{S}_\texttt{i}) \leq \min(n, m)$ means that $\texttt{S}_\texttt{i}$ must cover at least one n-level subtree The condition 1 ensures the number of subspaces which receive the message are not greater than 2^U.

The recipient node. The recipient node, i.e. the node which has received the $\texttt{REPLICATE}(\texttt{message}, \Delta_{\texttt{S}_\texttt{i}}, \Delta_\texttt{i})$ request belongs to an n-level subtree to which that message was sent to. This node is also responsible for broadcasting the message further, to all n-level subtrees covered by Δ_{S_i} which do not belong to the subspace ranging from Δ_i to $2 \cdot \Delta_i$, i.e. the subspace cover the subtree which contains the sent node in the case that the recipient space contains the sent node. The recipient node calculates $\delta_{\texttt{S}_\texttt{i}}$ of the space $\texttt{S}_\texttt{i}$ for which it is responsible from using Δ_{S_i} using following formula:

$$\delta_{\texttt{S}_\texttt{i}} = (m + n) - log_2 \Delta_{S_i}$$

If $\delta_{\texttt{S}_\texttt{i}} \leq \texttt{U} \cdot (\texttt{V} - \texttt{1})$, the node does the following:

1. Sends the request $\texttt{REPLICATE}(\texttt{message}, \Delta_{\texttt{S}_\texttt{j}}, \texttt{0})$ to all subspaces $\texttt{S}_\texttt{j}$ which satisfy the following conditions:
   - $\delta_{\texttt{S}_\texttt{j}} \leq \min(\texttt{n}, \delta_{\texttt{S}_\texttt{i}} + \texttt{U})$
   - $\texttt{S}_\texttt{j}$ does not contain the node.
   - $\texttt{S}_\texttt{j}$ does not have any children $\texttt{S}_\texttt{k}$ for which the first condition is satisfied.

   with $\delta_{\texttt{S}_\texttt{j}}$ is the depth of the subspace $\texttt{S}_\texttt{j}$ and $\Delta_{\texttt{S}_\texttt{j}}$ is the range of $\texttt{S}_\texttt{j}$.
2. Sends the request $\texttt{REPLICATE}(\texttt{message}, \Delta_{\texttt{S}_\texttt{j}}, \Delta_\texttt{j})$ to its closest n-level subtree of subspace $\texttt{S}_\texttt{j}$ that satisfies the following three conditions:
   - $\delta_{\texttt{S}_\texttt{j}} \leq \min(\texttt{n}, \delta_{\texttt{S}_\texttt{i}} + \texttt{U})$
   - $\texttt{S}_\texttt{j}$ contains the node.
   - $\texttt{S}_\texttt{j}$ does not have any children $\texttt{S}_\texttt{k}$ which satisfy the first condition.

with $\delta_{\mathtt{S_j}}$ is the depth of the subspace $\mathtt{S_j}$ and $\Delta_{\mathtt{S_j}}$ is the range of $\mathtt{S_j}$. $\Delta_j = 2 \cdot \Delta_i$.

3. Sends the $\mathtt{REPLICATE}(\mathtt{message}, \Delta_{\mathtt{S_j}}, 0)$ request to all subspaces $\mathtt{S_j}$ in the space ranging from 0 to Δ_{S_j} except the subspace ranging from Δ_i to $2 \cdot \Delta_i$ that satisfies the condition:

   - $\delta_{\mathtt{S_j}} \leq \mathtt{n}$

   with $\delta_{\mathtt{S_j}}$ is the depth of the subspace $\mathtt{S_j}$ and $\Delta_{\mathtt{S_j}}$ is the range of $\mathtt{S_j}$.

If $\delta_{\mathtt{S_i}} > \mathtt{U} \cdot (\mathtt{V} - \mathtt{1})$, the node does the following:

1. Sends the request $\mathtt{REPLICATE}(\mathtt{message}, \Delta_{\mathtt{S_j}}, 0)$ request to all subspaces $\mathtt{S_j}$ in the space ranging from 0 to Δ_{S_j} except the subspace ranging from Δ_i to $2 \cdot \Delta_i$ that satisfies the condition:

   - $\delta_{\mathtt{S_j}} \leq \mathtt{n}$

   with $\delta_{\mathtt{S_j}}$ is the depth of the subspace $\mathtt{S_j}$ and $\Delta_{\mathtt{S_j}}$ is the range of $\mathtt{S_j}$.

The above process finishes once all of the n-level subtrees have received the message.

Example. Consider an 7-bits identifier overlay in which 5 first bits of identifiers are used for representing the `netID` part, i.e. $n = 5$. Assume that the overlay is fully occupied by peers and let $U = 3$ and $V = 1$. The range of whole distance space is $\Delta_{S_0} = 2^7$. The distance covered by one n-level subtree is $\Delta = \frac{\Delta_{S_0}}{2^5}$. The FOGP peer, denoted by `A`, with the `ID` of 0000000 broadcasts a message to all n-level subtrees. The broadcast process happens as following:

Node A**.** The routing table of node `A` is illustrated in the Figure 3.9. In the routing table of `A`, 8 equals subspaces numbered from 0 to 7 at Depth 3 are satisfied conditions:

1. $\delta_{\mathtt{S_i}} \leq \min(\mathtt{5}, \mathtt{3})$;
2. $\mathtt{S_i}$ does not have any children $\mathtt{S_j}$ which satisfy the previous condition;

Therefore, `A` sends the request $\mathtt{REPLICATE}(\mathtt{message}, \Delta_{\mathtt{S_i}}, \Delta_{\mathtt{i}})$ with $\Delta_{S_i} = \frac{\Delta_{S_0}}{2^3}$ and $\Delta_i = 0$ to 7 random nodes in the subspaces numbered from 1 to 7. None of those spaces contains the node A. Each recipient node is responsible for broadcast the message to n-level subtrees in the distance from the node itself to $\frac{\Delta_{S_0}}{2^3}$. Assume that node B, belonging to subspace numbered 5, is one of those recipient nodes. The operation performed by other recipient nodes is the same as the one performed by node B.

For the subspace numbered 0, i.e. contains the node A, node A sends the request $\mathtt{REPLICATE}(\mathtt{message}, \Delta_{\mathtt{S_i}}, \Delta_{\mathtt{i}})$ with $\Delta_{S_i} = \frac{\Delta_{S_0}}{2^3}$ and

$$\Delta_i = \Delta = \frac{\Delta_{S_0}}{2^5}$$

to the n-level subtree which is closest to A, i.e. the subspace numbered 1 at Depth 5. Assume that C is the recipient node.

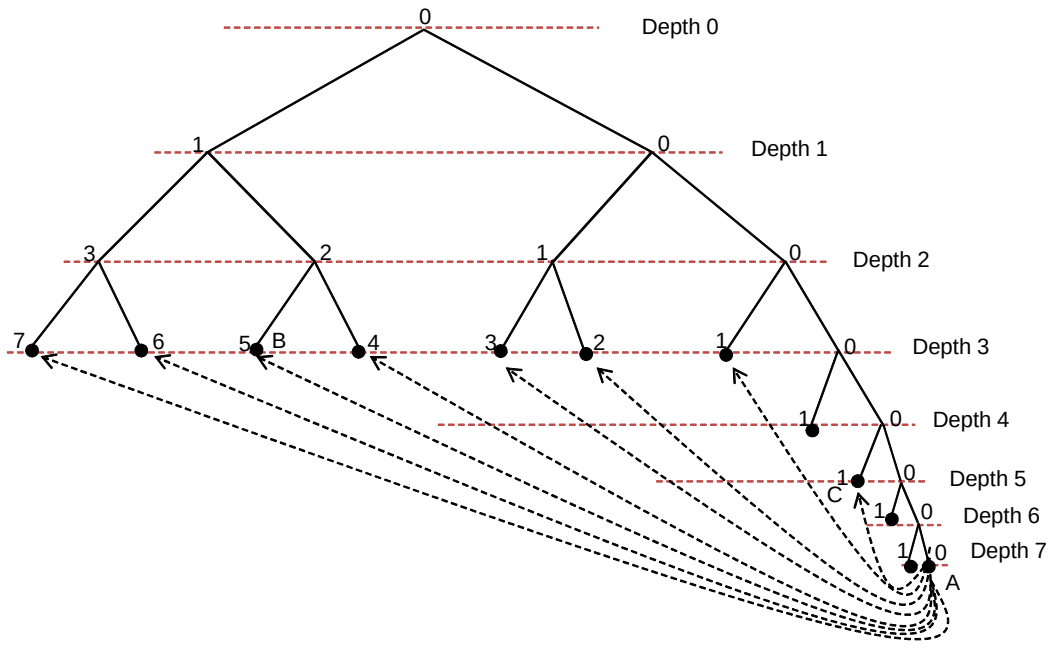

Figure 3.9: Operation performed by node A

Node B**.** Because the overlay is fully occupied, the routing table of node B is similar to the routing table of node A. The operation performed by node B is illustrated in the Figure 3.10a. Node B is responsible for broadcast the message to n-level subtree in the space ranging from 0 to $\frac{\Delta_{S_0}}{2^3}$. Because the depth of this space is 3 and $3 > U \cdot (V-1)$ with $U = 3$ and $V = 1$, node B sends the request

$$\texttt{REPLICATE}(\texttt{message}, \frac{\Delta_{S_0}}{2^4}, 0)$$

request to the subspace numbered 1 at Depth 4 and the request

$$\texttt{REPLICATE}(\texttt{message}, \frac{\Delta_{S_0}}{2^5}, 0)$$

request subspace numbered 1 at Depth 5. Those subspaces are the ones that belongs the space ranging from 0 to $\frac{\Delta_{S_0}}{2^3}$ and have the depth less than 5.

Assume that E and D are recipient nodes belonging to recipient subspaces at Depth 4 and Depth 5 respectively.

Upon receive the request from node B, node D stops forwarding the request because the depth of subspace that it is responsible for is 5, i.e. not less than n with $n = 5$.

Node E is responsible for broadcast the message in a subspace at Depth 4, i.e. less than 5 therefore it forwards the request to node F which belonging to the subspace numbered 1 at Depth 5 in the routing table of node E, see Figure 3.10b.

Similar to node D, node F stops forwarding the message.

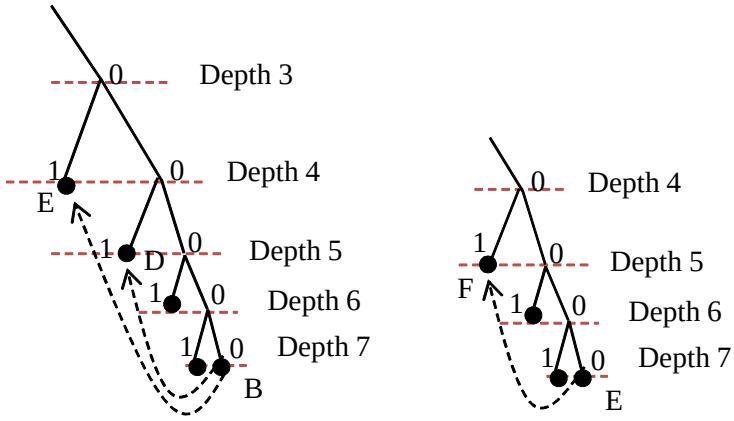

Figure 3.10: a. Operation performed by node B b. Operation performed by node E

Node C.

The operation performed by node C is illustrated in the Figure 3.11a. Node C is responsible for broadcast the message to n-level subtree in the space ranging from 0 to $\frac{\Delta_{S_0}}{2^3}$ except the subspace ranging from Δ to $2 \cdot \Delta$ with $\Delta = \frac{\Delta_{S_0}}{2^5}$. Because the depth of this space is 3 and $3 > U \cdot (V - 1)$ with $U = 3$ and $V = 1$, node B sends the request

$$\texttt{REPLICATE}(\texttt{message}, \frac{\Delta_{S_0}}{2^4}, 0)$$

request to the subspace numbered 1 at Depth 4 which is the one that belongs the space ranging from 0 to $\frac{\Delta_{S_0}}{2^4}$ and has the depth less than n with $n = 5$. Assume that G is the recipient node.

Similar to node E, upon receive the replicate request, node G forwards the request to node H which stops forwarding the message, see Figure 3.11b.

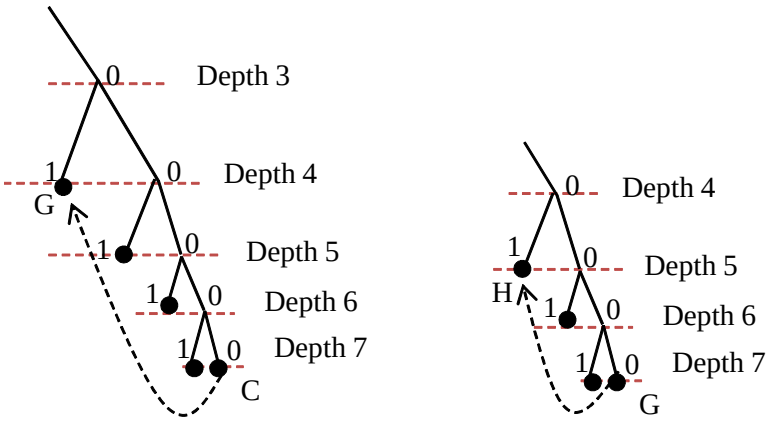

Figure 3.11: a. Operation performed by node C b. Operation performed by node G

Discussion and Analysis. Similar to the unicast algorithm, the message also touches only one node per subtree in Broadcast scheme. Thus the n-level subtrees can be seen as

virtual nodes in the n-bits overlay. In the first V rounds of sending message, the range of the space one recipient node is responsible for broadcasting the message reduces at least 2^U per round and reduces at least twice per round after that. If $log_{2^U} K < V$, after $log_{2^U} K$ rounds of sending request, i.e. message traverses through $log_{2^U} K$ hops, the range of the space a node is responsible for broadcast is:

$$\frac{2^n}{2^{U \cdot log_{2^U} K}} = \frac{2^n}{K}$$

The number of n-level subtrees belonging to this space is 1 with high probability. This subtree is the subtree containing the recipient node. Thus it takes $O(log_{2^U} K)$ hops to broadcast the message to all n-level subtrees in this case. Otherwise, if $log_{2^U} K > V$, after V hops, the range of the space a node is responsible for broadcast is $\frac{2^n}{2^{U \cdot V}}$. The number of n-level subtrees stay in this space is:

$$\frac{\frac{2^n}{2^{U \cdot V}}}{\frac{2^n}{K}} = \frac{K}{U \cdot V}$$

With the same analysis, after

$$log_2 \frac{K}{U \cdot V}$$

hops more, the message touches all subtrees with high probability. Therefore, if $log_{2^U} K > V$, the message is broadcasted to all n-level subtrees after

$$V + O(log_2 \frac{K}{U \cdot V})$$

hops.

If the routing table of nodes are full, the broadcast tree is balance to depth V and from the root, i.e. the initiator node, to the nodes in depth V, each node has 2^U children. This is important characteristic of n-level broadcast algorithm which makes the load on nodes quite balance and makes the algorithm robust to churn because each node is only responsible for a small part of the whole distance.

OGP broadcast tree. The broadcast tree of n-level broadcast algorithm in above example illustrated in Figure 3.12. In summary, the OGP peer A is the initiator node. It sends the messages to $2^U = 2^3 = 8$ other nodes. These nodes then forward the message to 15 other nodes. Each of them either stops forwarding the message or sends the message to another node.

We expect the n-level broadcast algorithm to achieve scalability, resilient to churn and load balance. Scalability means the algorithm scales well with the increase of participants in terms of number of messages passing in the algorithm and the amount of overheads for maintaining the broadcast tree. Resilient to churn means the join/leave of participants does not seriously affected to the algorithm. Load balance means the work load is evenly distributed over the nodes at each depth of broadcast tree. The work load here means the size of identifier space that a node is responsible for continuing to broadcast the message.

Under the churn condition, the nodes, which are closer to the root in the broadcast tree, play more important role than the nodes which are farther to the root. This because

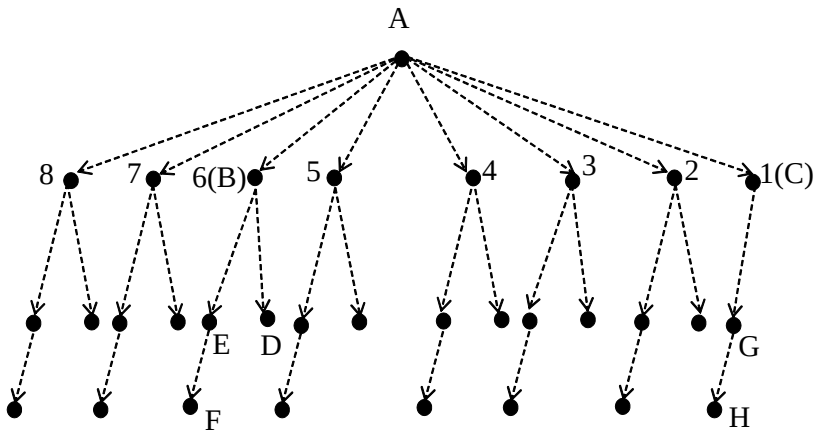

Figure 3.12: Broadcast tree routed from FOGP node A

the closer nodes are the roots of larger subtrees, i.e. are responsible for broadcast the message to larger sizes of identifier spaces. This means that in the broadcast tree, the resilient to churn and load balance of the nodes at depth p are more important than those of nodes at depth q with $p < q$.

In OGP, by adding more pointers to several farther spaces in routing table of nodes, we keeps the broadcast tree balance to depth V and each node at depth p with $p < V$ is responsible for broadcast the message in the identifier space of size

$$\frac{\Delta_0}{2^{U*p}}$$

with Δ_0 is the size of whole identifier space. This makes the OGP broadcast tree balance to the depth V. On the other hand, we can increase the U parameter to increase the resilient to churn.

In OGP broadcast, the number of messages passing only depends on the number of n-level subtree while does not depend on the number of nodes. To broadcast a message in an overlay with k subtrees, the algorithm need only $(k-1)$ messages passing. This means the our broadcast tree achieves the scalability.

The adding of more pointer to a number of farther subtree increase the amount of maintenance overheads. However, we can control these additional overheads by tuning the U and V parameters.

3.3.1.8 n-level multicast.

Algorithm. n-level multicast is an algorithm used by a FOGP node to send a message to a group of n-level subtrees on the OGP overlay of which it is not a member. This group is called the *multicast group*. The multicast group can be formed in different ways, e.g. by all n-level subtrees at distance Δ from the current node, by all n-level subtrees with given n-bit prefixes, or by all n-level subtrees with n-bit prefixes belonging to a given range.

The multicast algorithm is similar to the broadcast algorithm with the following general idea: a node is responsible for multicasting the message within a certain distance space. The multicast process happens as following:

The initiator node checks to see if there is overlap between the multicast group and the following spaces in its routing table:

1. The subspaces that each of which corresponds to one k-bucket and covers at least one n-level subtree.

2. The subspaces together cover one n-level subtree and each of which are further divided. corresponds to one k-bucket are represented by the subspace covers that

For each subspace that overlaps with the multicast group, if the routing table contains a contact belonging to both the subspace and the multicast group, that contact is chosen. Otherwise, the node chooses a contact belonging to that subspace which is closest to the multicast group, i.e. the node whose n-bit prefix is closest to n-bit prefix of one of subtrees belonging to the multicast group.

The initiator node then sends the message to the chosen node and asks the chosen node to be responsible for multicasting the message to the n-level subtrees belonging to both that subspace and the multicast group.

The above process continues until all n-level subtrees in the multicast group have received the message.

Discussion and Analysis. Using the same analysis with broadcast algorithm discussion, if $log_{2^U} K < V$, it takes $O(log_{2^U} K)$ hops to multicast the message to the destination n-level subtrees. Otherwise, it takes

$$V + O(log_2 \frac{K}{U \cdot V})$$

hops.

3.3.1.9 Summary.

The routing cost in three OGP routing algorithms, illustrated in Tables 3.1, are the same and smaller than $O(log_2 K)$. We notice that the routing cost only depend on the number of n-level subtrees and doesn't depend on the number of FOGP nodes.

Algorithms	Routing cost
n-level unicast	$O(log_{2^U} K)$ if $log_{2^U} K < V$
n-level broadcast	$V + O(log_2 \frac{K}{U \cdot V})$, if $log_{2^U} K > V$
n-level multicast	

Table 3.1: The cost of routing in OGP routing algorithms.

3.3.2 Lightweight OGP protocol

The lightweight OGP protocol is performed by LOGP peers to communicate with FOGP peers. A LOGP peer maintains a routing list, which is a fixed-size list containing information about some FOGP peers in the OGP overlay by periodically asking for the routing table of FOGP peers in its routing list and then using the information in these routing table for updating the routing list. At the bootstrap time, a LOGP peer known some bootstrap FOGP peers via external mechanisms such as from websites. A LOGP peer sends messages to standard overlays of which it is not member by simply sending these messages to the first FOGP peer in its routing list which will forward its messages.

3.3.3 Cooperation application

We put the figure which illustrates the structure of a FOGP peer (Figure 3.1) here for the sake of clarity.

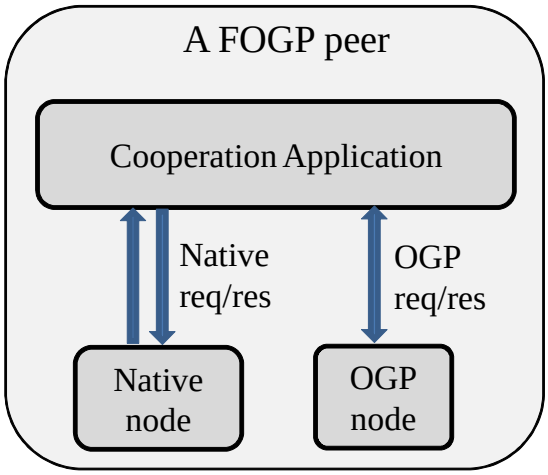

Figure 3.13: A FOGP peer

The cooperation application in a FOGP peer is built on top of the OGP routing layer, and is responsible for following tasks:

- Launching the delivery of requests to P2P systems via the Native node or the OGP node or both and receiving results from these nodes.
- Transcoding of messages and data between formats of P2P systems and the intermediary formats defined by itself at interface level.
- Communication with each other.

The first and the third tasks can be achieved easily. The intermediary formats, in the second task, vary from application to application. Therefore, we cannot introduce a common intermediary formats. In the next chapter, we introduce the IFP protocol, running on the top of OGP, which is an application allowing heterogeneous P2P file-sharing networks to cooperate, together with the respective intermediary formats.

3.4 Evaluation of Protocols

3.4.1 Evaluation metrics

We characterized the OGP protocol by three metrics $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$. Here, $\mathcal{M}_1$ is the success ratio for requests sent from a FOGP peer to the standard overlay containing the requested data and then back to the originator. $\mathcal{M}_2$ is the number of hops in OGP overlay that a request passed in the successful routing, i.e. the routing cost while $\mathcal{M}_3$ is the bandwidth generated by OGP protocol in a peer per minute, i.e. the cost for running OGP.

Similar to OGP protocol, the lightweight OGP protocol is evaluated by the three following metrics $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$. Here, $\mathcal{N}_1$ is the success ratio of request sent to a

FOGP peer and the corresponding response is turned back, $\mathcal{N}_2$ is the number of hops the request sent from a LOGP peer traversed to reach a FOGP peer, i.e. the routing cost, and finally $\mathcal{N}_3$ metric represents traffic generated by a LOGP peer in a given period of time, to discover and maintain its list of FOGP peers.

3.4.2 Experimental setup

To evaluate the OGP and lightweight OGP protocols, a complete system, in which the OGP overlay is used to interconnect different standard overlays, has been has been implemented and deployed in large scale infrastructure. The experiments consisted in testing the lookup of random data distributed across all of the standard overlays, with each piece of data unique. The FOGP and LOGP peers periodically looked up a random piece of data on any of the standard overlays. The FOGP peers used the n-level broadcast algorithm for the request broadcast which is the most comprehensive scenario. JOGP, a simple lookup protocol based on OGP and lightweight OGP, has been developed and implemented in Java for the experiments. Our experiments were performed on the French Grid5000 platform [G5k], which aims at providing a nation-wide testbed to study large scale parallel or distributed systems.

There are two experimental scenarios. The first evaluates the metrics of the OGP protocol while the second for evaluating the metrics of the lightweight OGP protocol when performing data lookup with several kind of standard overlays. In both scenarios, twenty networks of various types - Chord, Kademlia, and Gnutella- are deployed. Each network constitutes 50 nodes. In the first scenario, there are only FOGP peers and the percentage of FOGP peers varies from 6% to 40% while in the second scenario, the percentage of LOGP peers varying from 10% to 60% with the percentage of FOGP peer is set to 10% and 20%. The churn of nodes in all scenarios follows the Pareto distribution. We evaluate the protocols in high churn environment which is expected to see in near future. In the first scenarios, the lifetime mean of nodes is set to 900, while it is set to 1800 seconds in the second scenario. The parameters of the two scenarios are given in Table 3.2.

Experimental parameters	Scenario 1	Scenario 2
% of FOGP peers	6, 10, 20, 30, 40	10, 20
% of LOGP peers	0	10, 20, 40, 60
Lifetime mean (second)	900, 1800	1800
No. of overlays	20	
No. of nodes per overlay	50	
Type of overlays	Chord, Kademlia, Gnutella	

Table 3.2: Values of experimental parameters.

Each experiment includes 3 successive phases: *initial phase*, *stabilizing phase* and *evaluation phase*. In the initial phase, nodes are created and join overlays. After all nodes join the overlays, experiment goes to second phase, i.e. stabilizing phase in which nodes complete their routing table and the system becomes stable. The evaluation phase to collect statistics starts when the second phase finishes. The duration of each of two last phases is T with T is the sum of lifetime mean and deadtime mean of a node in that experiment.

Each experiment is run 5 times. Average values and corresponding standard deviations of the metrics are plotted in all of the figures illustrate the experiment results.

3.4.3 Software

The software for evaluating the OGP framework constitute twofold:

JOGP: a P2P node client, which implements following protocols: OGP, lightweight OGP, Kademlia, Gnutella and Chord. It can be instantiated as OGP node, lightweight OGP node, Kademlia node, Gnutella node and Chord node. Every kind of node provides PUT and GET functions.

Manager: the experiment manager, which performs following tasks:

1. It automatically deploys the inter-connected overlays in large scale on Grid5000 platform using the configuration read from configuration file (a text file). In the configuration file, user can define various experiment parameters such as the number of overlays, the protocol of each overlay, the percentage of OGP, LOGP nodes, the duration time of the experiment, the time interval for lookup data, etc.

2. It launch the processes of PUT and GET data of random nodes using the configuration read from configuration file.

3. It collects statistics of the experiments such as the success rate of GET operation, the number of hops of successful GET operations, the numbers of messages generated by each node, etc.

The software are available on the website: http://www-sop.inria.fr/members/Giang.Ngo_Hoang/software.html

3.4.4 Experimental results

3.4.4.1 Lookup efficiency

In this experiment, we evaluate the success rate of operations performed by the FOGP peers, i.e. the $\mathcal{M}_1$ metric and the success rate of operations performed by the LOGP peers, i.e. the N_1 metric.

The OGP protocol. The results of the $\mathcal{M}_1$ metric is illustrated in Figure 3.14. The "Lifetime mean of 900s" and "Lifetime mean of 1800s" lines represent the $\mathcal{M}_1$ metric in the lookup system with the lifetime mean of nodes is set to 900s and 1800s, respectively.

From Figure 3.14, we see that the two lines share mostly the same shape and the line "Lifetime mean of 1800s" stays above the line "Lifetime mean of 900s". The "Lifetime mean of 1800s" and "Lifetime mean of 900s" lines dramatically increase from 77% to 97% and from 68% to 94% respectively with the increase of the percentage of FOGP peers from 6% to 10%, but only slightly vary in the range from 97% to 99% and 94% to 98% respectively when the percentage of FOGP peers is increased from 10% to 30%. The standard deviations of two lines are about 3 as the percentage of FOGP peers is set to 6% and stay in the range from 1.5 to 2 as the percentage of FOGP peers is increased from 10% to 30%

Discussion and analysis. The reason for the trend of values of $\mathcal{M}_1$ metric is as follows: with 6% of FOGP peers, the number of FOGP peers per overlay is $6\% \cdot 50 = 3$, meaning that there are 3 gateways to enter each standard overlay. In a high churn environment,

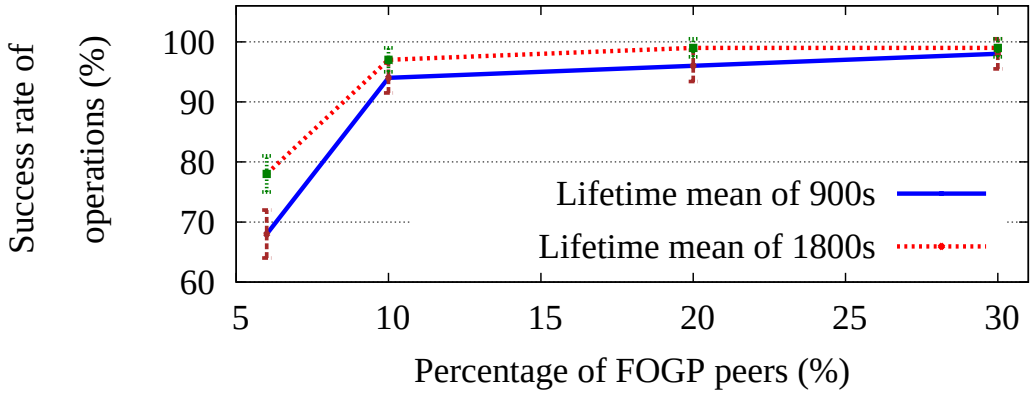

Figure 3.14: Success rate of operations of a FOGP node

some gateways can go down for at certain time. During this time, some other FOGP peers do not have any backup gateways for the downed gateways in their routing tables, as the number of gateways to enter an standard overlay is only 3. With 10% of FOGP peers, there are $10\% \cdot 50 = 5$ gateways to enter a standard overlay, and this number appears to be sufficient for the FOGP peers to build their routing table with quite enough backup. As the percentage of FOGP peers rises, values of $\mathcal{M}_1$ slightly increase, since the routing table of the FOGP peers has more backup.

The fact that values of $\mathcal{M}_1$ varies from 94% to 98% and 97% to 99% when the percentage of the FOGP peers is $\geq 10\%$ in cases the lifetime mean is set to 900s and 1800s respectively shows the efficiency of the OGP protocol. If a piece of data exists in a standard overlay, a FOGP peer will reach that overlay and the found data will be forwarded back to that FOGP peer with a 94%-98% and 97%-99% probability respectively.

The low standard deviation when the percentage of the FOGP peers is $\geq 10\%$ shows the stable of OGP routing.

In higher churn rate, i.e. nodes join and leave the overlays more frequently, the probability of a contact in routing table of an FOGP node become unavailable is higher, then the probability of this entry fails to perform the OGP broadcast is higher. Therefore, the values of $\mathcal{M}_1$ metric is lower in higher churn rate environment. That why the line "Lifetime mean of 1800s" stays above the line "Lifetime mean of 900s".

The lightweight OGP protocol. Figure 3.15 illustrates the values of the $\mathcal{N}_1$ metric. The two lines "FOGP peers of 10%" and "FOGP peers of 20%" represent values of $\mathcal{N}_1$ metric with 10% and 20% of FOGP peers, respectively. From the figure we can see that the two lines vary from 99% to 100% when the percentage of LOGP peers comes from 10% to 60%. The standard deviation are only 1 for all cases.

Discussion and analysis. The results show that when a LOGP peer performs a lookup, the percentage of requests successfully sent to FOGP peers, and the percentage of found data successfully sent back to that LOGP peer is nearly 100%, i.e. highly efficient.

3.4.4.2 Routing cost

In this section, we evaluate the $\mathcal{M}_2$ metric of the OGP protocol, i.e. the routing cost or the number of hops that the request traversed over the OGP overlay in a successful

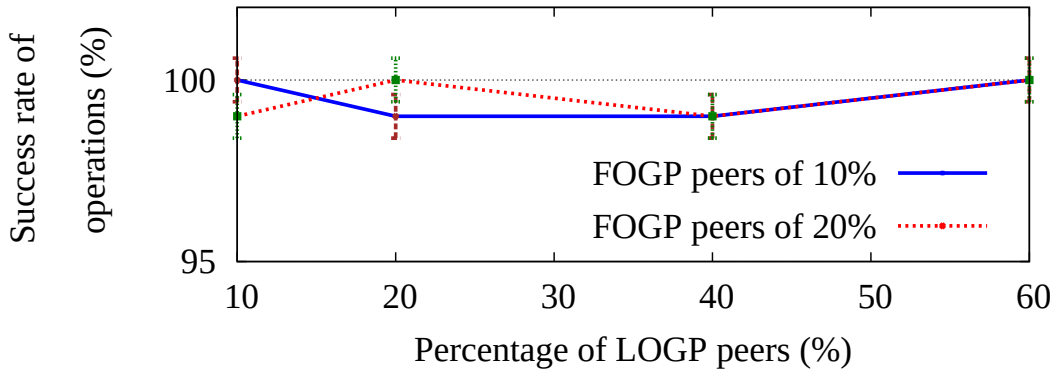

Figure 3.15: Success rate of operations of an LOGP node

lookup performed by a OGP peer. The $\mathcal{N}_2$ metric of the lightweight OGP protocol is always 1 hops because the lightweight OGP nodes directly send the request to the FOGP nodes. Figure 3.16 illustrates the $\mathcal{M}_2$ metric of a FOGP peer in cases the lifetime mean is set to 900s (the "Lifetime mean of 900s' line) and 1800s (the "Lifetime mean of 1800s' line).

We can see from the figure that the two lines both increase, from about 3.3 to about 3.6 as the percentage of FOGP peers rises from 6% to 10%, and then slightly vary in the range between 3.6 and 3.9 as the percentage of FOGP peers rises from 10% to 30%.

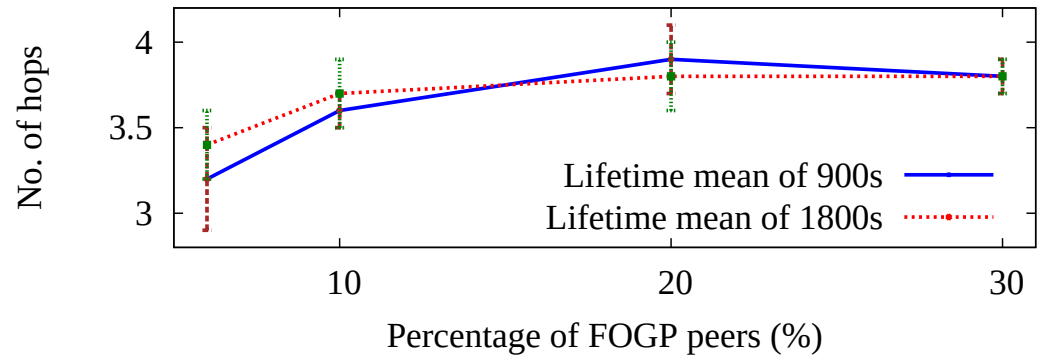

Figure 3.16: Routing cost of OGP broadcast algorithm

Discussion and analysis. The experiment results confirm our evaluation of routing cost. In our experiments, U=3, V=1 and K=20, thus the expected value of M_2 is $1 + O(log_2 20/3)$ or approximately 3.7 hops, i.e. a constant.

The experiment shown that the values of M_2 metric is approximately the expected constant when the percentage of FOGP nodes larger than 10% while smaller than expected constant with the 6% of FOGP. The reason is as following. In high churn rate environment, the routing with more hops fails at higher probability than the routing with less hops (each hop has a certain probability of failure). In our experiment, at 6% of FOGP peers, the ratios of success routing, i.e. M_1, are only 68% and 77% in cases the lifetime mean is set to 900s and 1800s respectively. This means the number

of routing with more hops which fails is considerably higher than the number of routing with less hops which fails. Hence the average hops of success routing, i.e. M_2, is lower than the expected constant. When the percentage of FOGP nodes increases from 10% to 30%, values of M_1 are near to 100% thus values of M_2 also approximately the expected constant.

3.4.4.3 Traffic

This section evaluates the traffic generated by a FOGP peer on OGP overlay ($\mathcal{M}_3$ metric) and a LOGP peer to maintain its OGP list ($\mathcal{N}_3$ metric) in a period of time.

The OGP protocol. Figure 3.17 shows the traffic generated by a FOGP peer on the OGP overlay during one minute when the percentage of FOGP peers increases from 6% to 30% (the "FOGP" line) and the traffic generated by a Kademlia node during one minute (the "Kademlia" line)

From Figure 3.17, we can see that the "FOGP" line rise from around 61 messages/node/minute to about 69 messages/node/minute and then around 97 messages/node/minute, as the percentage of FOGP peer increase from 6% to 10% and then to 30%, respectively. The figure also show us the "Kademlia" line is a horizontal line at around 58 messages/node/minute.

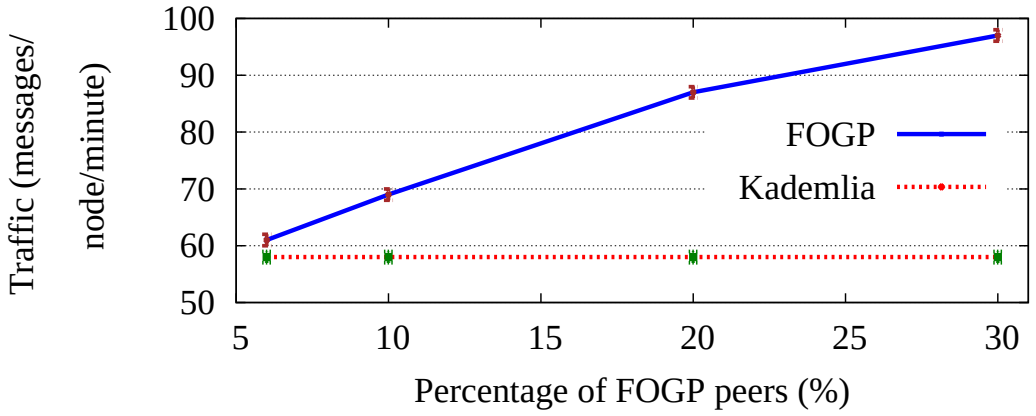

Figure 3.17: Traffic of a FOGP node on the OGP overlay.

Discussion and analysis. The traffic generated by an FOGP peer is increased as the percentage of FOGP peer increase from 6% to 30% since with the increase of the percentage of FOGP peers per overlay, the size of the OGP overlay increases (from 60 nodes to 300 nodes) and so does the traffic for maintaining it. In our scenario, as shown in the evaluation of M_1 metric, OGP achieves high exhaustiveness with only 10% of peers are FOGP peer, at this point the traffic generated by a FOGP node is about 20 % higher than the traffic generated by a Kademlia node. This shows the efficient of OGP protocol in term of generated traffic.

The lightweight OGP protocol. Traffic generated by a LOGP peer for maintaining its FOGP peer list (the $\mathcal{N}_3$ metric) as the percentage of LOGP peer increase from 10% to 60% is illustrated in Figure 3.18. The "LOGP" and "Kademlia" lines represent the traffic generated by a LOGP node and a Kademlia node in a minutes respectively. Both lines

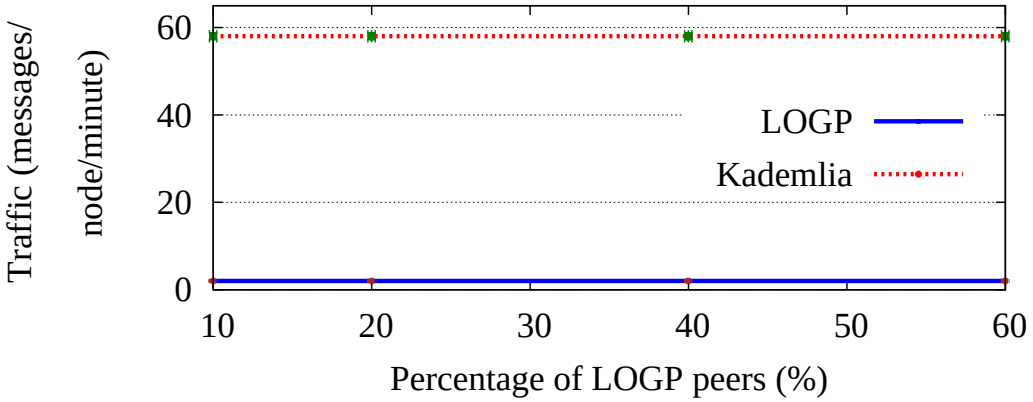

Figure 3.18: Traffic of a LOGP node.

are horizontal lines at values of 2 messages/node/minute and 58 messages/node/minute respectively.

Discussion and analysis. LOGP peers generate little traffic for maintaining its routing list (around 2 messages/node/minute), which also does not depend on the percentage of FOGP peers and the percentage of LOGP peers in the lookup system. This feature proves the scalability of inter-overlay lookup system with the participation of LOGP peers.

3.4.5 Related works

As discussed in the literature review, the largest distinguish feature of the cooperation model introduced by current work, comparing with other models, is the decoupling of inter-overlay routing function and application bridging function.

In this section, we compare the inter-overlay routing solution proposed by current work with the inter-overlay routing solutions proposed by other studies based on following dimensions: the types of applicable overlays, the back-ward compatible ability, the scalability, the routing scheme.

Applicable overlays. OGP allows for inter-routing over *heterogeneous* overlays, including both structured and unstructured overlays.

Previous works manage inter-routing between either only DHTs, i.e. structured overlays [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012], or only unstructured overlays [Konis at al 2006].

Backward compatibility. OGP guarantees *backward-compatibility*, i.e. it can function with already deployed overlays without any changes being made to the deployed peers of the existent overlays.

[Konis at al 2006] and black box version of Synapse in [Liquo at al 2010, Cianc et al 2011, Cianc at al 2012], enable the backward-compatible routing between overlays. However, the black box model of Synapse in [Liquo at al 2010] was not evaluated at all while [Konis at al 2006] is only for unstructured overlays. Other works, including [Cheng 2007, Liquo at al 2009] and white box version of Synapse in [Liquo at al 2010] require modification of all peers.

Scalability. OGP features *better control over routing*, by namely allowing for choosing to broadcast the message to all overlays, or multicast the message to a group of overlays, or simply route the message to a specific overlay. In all cases, an overlay receive only one copy of the message, without duplication. This feature makes OGP scalable and fundamentally different from previous works, where there was no control on which overlays will receive the query and, mostly, a query could reach an overlay multiple times, triggers numerous duplicated lookup processes while not reaching some other overlays at all.

Routing scheme. OGP based on deterministic routing, i.e. a peer can reach all overlays of which it is not member. In other works, the queries are forwarded only between neighbor overlays, i.e. overlays share co-located nodes [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Cianc et al 2011, Cianc at al 2012] or logical links [Konis at al 2006]. There, the overlays that a query can reach are limited by a TTL value to avoid looping.

As a matter of fact, except the black box version of Synapse in [Cianc et al 2011, Cianc at al 2012], hereby called Synapse for sake of brevity, which is also proposed by us and closest to current solution as well as was evaluated using the same Grid5000 platform, we didn't compare this work with other works including [Cheng 2007, Liquo at al 2009, Liquo at al 2010, Konis at al 2006] either due to lack of information (black box model of Synapse in [Liquo at al 2010]) or because this work is different from them in the main features (the rest).

However, the evaluation of Synapse was limited. The exhaustiveness provided in [Cianc at al 2012] depend on the lookup efficiency of native protocols which was not provided while the routing cost as well as the bandwidth generated by Synapse nodes was not evaluated. According to [Cianc at al 2012], a Synapse node must simultaneously belong to at least 4 standard overlays for achieving high exhaustiveness while 10% of peers are FOGP peer is enough for achieving high exhaustiveness in current work. Hence we compare the two works in terms of amount of generated traffic for achieving high exhaustiveness via analysis.

The Figure 3.19a compares the traffic generated by a FOGP node, a LOGP node and a Synapse node. The unit of y-axis is the average traffic generated by a native node. As the OGP overlay is basically a DHT with some more entries in the routing table, the traffic of a FOGP node, including traffic of a native node and traffic of an FOGP node generated in OGP overlay, is about twice the traffic of one native node. The traffic of an LOGP node is approximate the traffic of a native node because the lightweight OGP protocol itself generates very few messages. The traffic of a Synapse node includes traffic of 4 native nodes and traffic generated by Synapse protocol which was not evaluated via simulation or experiment.

In current solution, only a small number of peers are FOGP peer, other peers are LOGP peer while in [Cianc at al 2012], any peer wish to take full advantage of inter-routing must be Synapse. Consider an example scenario performed in our experiment, 20 overlays with total 1000 nodes, in current solution 10% of peers are FOGP and 60% of peers are LOGP while in solution proposed by [Cianc at al 2012], 70% of peers are Synapse. The Figure 3.19b show the overall traffic generated by both FOGP nodes and LOGP nodes in our solution and Synapse in solution proposed by [Cianc at al 2012] in the example scenario. The unit of y-axis is the average traffic generated by a native

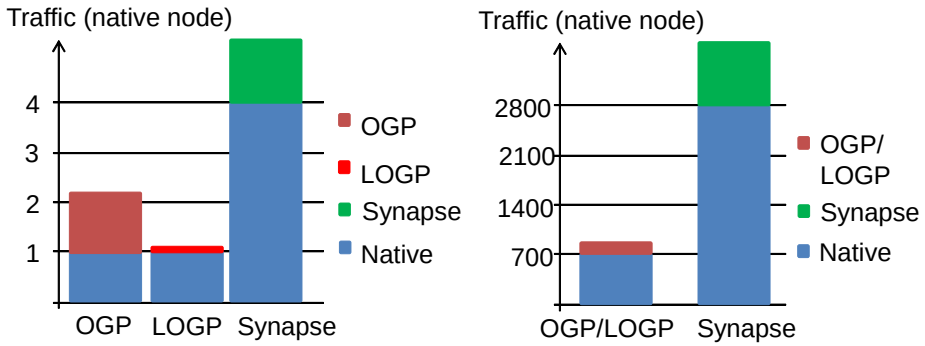

Figure 3.19: (a) Traffic of a node of compared protocols (b) Traffic of all nodes of compared protocols

node. The traffic generated by 100 FOGP nodes (10% of peers) and 600 LOGP (60% of peers) equals the traffic generated by 700 native nodes plus the traffic generated by 100 OGP peers in OGP overlay, i.e. approximately traffic generated by 800 native nodes. The traffic generated by 700 Synapse nodes equal traffic generated by $700 \cdot 4 = 2800$ native nodes plus the traffic generated by Synapse protocol itself in 700 Synapse nodes. This is much higher than traffic generated by FOGP and LOGP in current solution.

3.5 Chapter summary

In this chapter, we have introduced an efficient model for backward-compatible cooperation of heterogeneous P2P systems. The model consists of the OGP framework for inter-overlay routing and the cooperation application on top of the OGP framework, mapping the interface of these P2P systems to a mediatory interface.

The evaluations of the OGP protocol indicate that having only 10% of peers as FOGP peers is sufficient for achieving a high success ratio of round trip inter-overlay routing operations in high churn conditions: more than 94% and 97% success ratio is achieved when the lifetime mean of a node is 900s and 1800s, respectively. Both FOGP and LOGP peers are proven to be efficient in terms of the number of hops needed for data lookup. LOGP peers need only one hop to reach FOGP peers while the number of hops a lookup traversed over the OGP overlay only depends on the number of overlays inter-connected by the OGP overlay. The experiments also show that a LOGP peer generates the traffic nearly as same as that generated by a blind peer. This, coupled with control over the routing between standard overlays makes our inter-overlay routing solution scalable.

In the next chapter, we will develop a complete solution for cooperation of P2P file-sharing networks.

This chapter relies on following conference papers:

1. Giang Ngo Hoang, Luigi Liquori and Hung Nguyen Chan. Backward-Compatible Cooperation of Heterogeneous P2P Systems. The 15th International Conference on Distributed Computing and Networking, ICDCN 2014, accepted.

2. Giang Ngo Hoang, Luigi Liquori, Vincenzo Ciancaglini, Petar Maksimovic and Hung Nguyen Chan. A backward-compatible protocol for inter-routing over het-

erogeneous overlay networks. The 28th Annual ACM Symposium on Applied Computing, SAC '13, pp. 649-651, Coimbra, Portugal, March 18-22, 2013.

3. Vincenzo Ciancaglini, Luigi Liquori, Giang Ngo Hoang and Petar Maksimovic. An Extension and Cooperation Mechanism for Heterogeneous Overlay Networks. NETWORKING 2012 Workshops - International IFIP TC 6 Workshops, ETICS, HetsNets, and CompNets, Held at NETWORKING 2012, pp. 10-18, Prague, Czech Republic, May 25, 2012.

4. Vincenzo Ciancaglini, Luigi Liquori, Giang Ngo Hoang. Towards a Common Architecture to Interconnect Heterogeneous Overlay Networks. The 17th International Conference on Parallel and Distributed Systems, ICPADS 2011, pp. 817-822, Tainan, Taiwan, December 7-9, 2011.

Chapter 4

Backward Compatible Cooperation of Heterogeneous File-Sharing Networks

Contents

4.1 Introduction

In this chapter, we develop a complete *cooperation solution* for cooperating heterogeneous P2P file-sharing networks using the cooperation framework presented in Chapter 3.

P2P file sharing networks provides their users two main operations including uploading and downloading files so that the users can share their files to other users in the same network, and get files from them, respectively. However, current P2P file sharing networks use incompatible protocols and mechanisms for these operations.

Concerning to uploading file operation, the peers hosting certain files can keep the information of these files locally as in a Gnutella network, or put the information to other peers, e.g. in a Kad network, or send the information to external websites such as in a BitTorrent network.

For the downloading file operation which is constituted by two phases, namely search and download, most of file sharing networks support the search capability while BitTorrent network, the current most widely used P2P file sharing network, doesn't support.

Moreover, the routing of search messages in the networks supporting search capability depend on the peculiar overlay networks which differ from each other in many aspects. For example, in topologies and routing algorithms, overlays such as Kademlia are based on structured topologies with deterministic routing, while overlays based on Gnutella are based on unstructured topologies and flooding routing. Overlay networks can also use various types of queries as well as different encoding messages algorithms.

The mechanisms used in the download phase are also different between networks. In some P2P file sharing networks such as BitTorrent, KAD, peers download the expected file from multiple sources at once while in others like Gnutella, the file is downloaded from a single source. The protocols for exchanging files used in the above P2P file sharing networks are different from each other, too.

As a matter of fact, these incompatibilities prevent the cooperation between different P2P file-sharing systems while cooperation of these networks promises many benefits. For example, the resource space that a user can reach is largely increased; the overall storage is significantly saved because the number of copy of a file is reduced; the content redundancy is easily to achieved.

In this chapter, we introduce a solution for co-operating heterogeneous P2P file-sharing networks which is applicable for all current P2P file-sharing networks. In our model, peers running OGP protocol or LOGP protocol, also run a protocol, named *Inter-network File-sharing Protocol* (IFP) which is designed for bridging different P2P file-sharing networks at application level and coordinating different components in a peer. The bridging between different networks is enabled by transcoding of necessary messages in these networks to common messages defined by IFP. The peers running OGP protocol form a super-overlay (the OGP overlay) serving as the routing infrastructure for communication between heterogeneous P2P file-sharing networks. The peers, running FOGP and LOGP protocols, in different P2P file-sharing network employs this routing infrastructure for communicating with each other.

The rest of this chapter is structured as follows: in Section 4.2 we briefly introduce the model for cooperation P2P file-sharing networks. Section 4.3 describes the IFP protocol while the evaluation of the model is introduced in Section 4.4. The related work is introduced in Section 4.5. Finally, in Section 4.6, we present our conclusions and outline future work.

4.2 System Overview

4.2.1 Peer classification

The classification of peers in the file-sharing cooperation model is an instance of the one in OGP framework and is as following:

Blind peers are peers that belong to only one P2P file sharing network, are not aware of the existence of additional protocols, and use only the protocols native to their network.

FOGP peers , simultaneously belong to one P2P file sharing network and the OGP overlay. They run following protocols: the native protocols of their P2P file sharing networks, the OGP protocol of the OGP overlay and the IFP protocol for coordinating the parts in the peer and exchanging data between different file-sharing networks.

LOGP peers , belong only to one P2P file sharing system, do not participate in the OGP overlay, but keep a list of OGP peers. They run following protocols: the native protocol of the P2P file sharing networks they belong to, and the lightweight OGP protocol for communicating with OGP peers and the IFP protocol for exchanging data between different file-sharing networks.

4.2.2 Structures of FOGP peers and LOGP peers

Structures of a FOGP peer is shown in the Figure 4.1

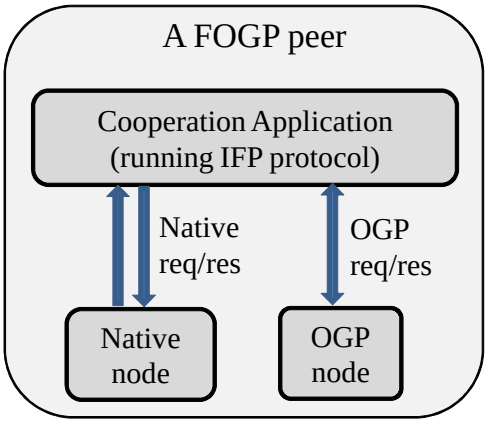

Figure 4.1: Structure of a FOGP peer

A FOGP peer has several components:

Native node is the peer of P2P file-sharing network that the FOGP belongs to. The Native node launches the requests on this P2P file-sharing network and returns the results to the Cooperation application which runs IFP protocol. The requests can be search request, download request or upload request while results can be search result, or downloaded file, etc.

OGP Node participates in the OGP overlay and provides unicast, multicast and broadcast routing cross P2P file-sharing networks for the FOGP node.

Cooperation application runs IFP protocol and performs following tasks:

1. Launching the search, download, upload processes on P2P file-sharing network of which the peer is member and receiving the results via the Native node.

2. Delivery of the search, download, upload request to P2P file-sharing networks of which the node is not member and receiving the results via the OGP node.
3. Communication and exchanging files with other FOGP, LOGP peers.
4. Defining the intermediary formats of messages and data and introducing the transcoding of messages and data between formats of P2P file-sharing networks and intermediary formats.

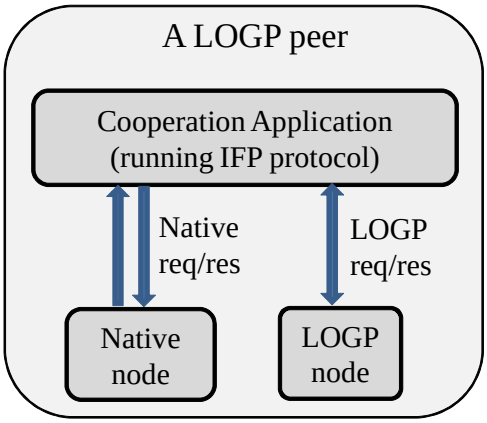

Figure 4.2: Structure of a LOGP peer

The structure of a LOGP peer, illustrated in Figure 4.2, is the same as the structure of a FOGP peer except that the LOGP Node which runs LOGP protocol replaces the OGP Node which runs OGP protocol.

4.2.3 Cooperation schemes

Users are provided two schemes of cooperation, namely inter-network downloading and inter-network uploading. While the inter-network uploading scheme allows the users to upload their files to any network, the inter-network downloading allows users to download files from any network.

Currently, the P2P file-sharing networks can be classified into two categories: searchable network and non-searchable network based on whether the network supports the search capability or not. Most of these networks today are searchable network which allow users to search file information which is located at peers in the network using the search protocol of that network. Examples of searchable networks are Kad, Edonkey and Gnutella. There is only one non-searchable network today, i.e. BitTorrent which requires the users to download the .torrent file from outside sources such as websites. We differentiate these kinds of networks in the cooperation schemes.

Following, we describe the two cooperation schemes.

4.2.3.1 Inter-network uploading

The processes of inter-network uploading in which the destination network is searchable and the destination network is non-searchable are illustrated in Figure 4.3(a) and Figure 4.3(b), respectively.

In the two figures, the green lines illustrate the communication following OGP protocol while the blue lines represent the communication following the standard protocol of P2P file-sharing network which is denoted by "Standard network". The communication performed by IFP protocol is illustrated by red lines. The case in which a peer uploads its file to the P2P file-sharing network of which the peer is member is trivial, thus is not shown in the figures for the sake of clarity.

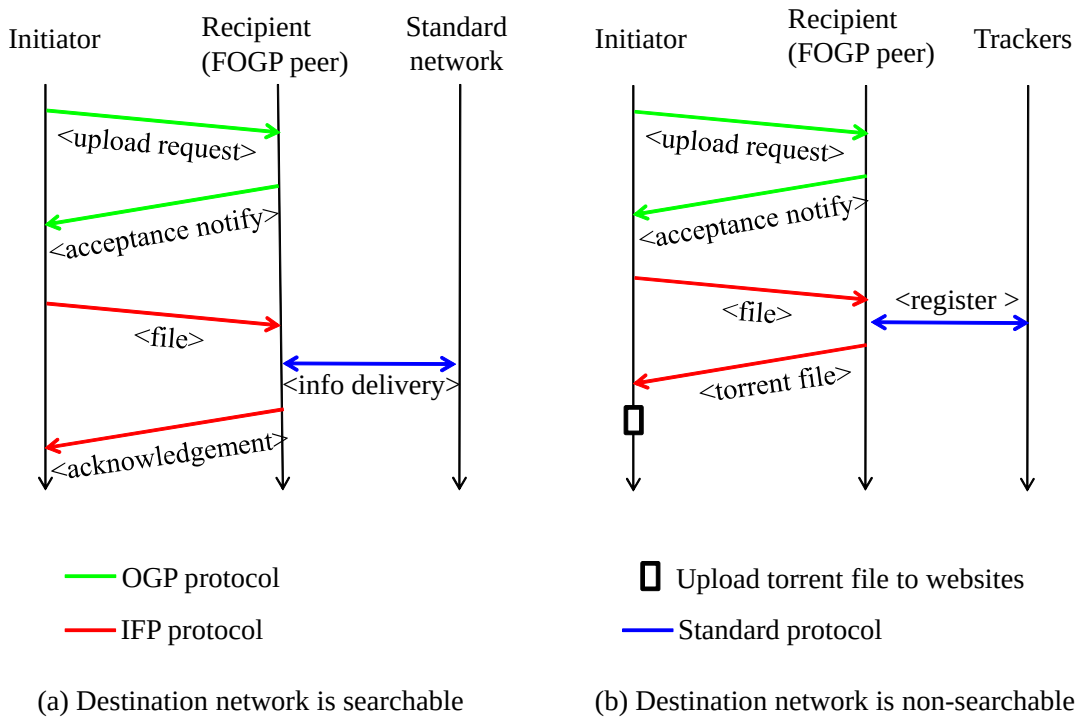

Figure 4.3: Inter-network uploading

Destination network is searchable. The uploading process for the case in which the destination network is searchable (see Figure 4.3(a)), happens as following.

Step 1. The *initiator*, which is a FOGP peer or a LOGP peer, sends upload requests, following OGP protocol via its OGP node or LOGP node respectively, to the *recipients* which are FOGP peers belonging to a group of networks to which the *initiator* wants to upload the file.

Step 2. Upon receiving the request, a *recipient* sends the response, following OGP protocol via its OGP node, notifying the *initiator* whether the upload request is accepted.

Step 3. Upon receiving the notification, if the upload request is accepted, the *initiator* peer sends the file to the *recipient* following the IFP protocol.

Step 4. The *recipient* then uploads the file to its standard network via its Native node which uploads the file using the uploading mechanism of that network. Upon finishing the upload operation, the *recipient* sends the acknowledgement back to the *initiator* following IFP protocol.

Destination network is non-searchable. The uploading process for the case in which the destination network is non-searchable, i.e. BitTorrent, (see Figure 4.3(b)) is different from the one for the case in which the destination network is searchable as following:

Step 4. Upon receiving the file from the *initiator*, the *recipient* create .torrent file for the file and register the .torrent file with some trackers. Then the *recipient* sends the torrent file back to the *initiator*.

Step 5. Upon receiving the .torrent file, the initiator uploads it to somewhere such as web sites.

4.2.3.2 Inter-network downloading

The inter-network downloading operation constitutes two phases. The first phase includes delivering of search request to a group of networks and then receiving of search results from these networks. The second phase is to obtain the searched file from certain network. Among two above phases, the first phase is enabled by OGP protocol while second phase is achieved by IFP protocol.

The inter-network downloading operation are illustrated in the Figure 4.4 and the Figure 4.5. The Figure 4.4 represents the scenario in which the destination network, i.e. the one contains the peer receiving the search request, is searchable while the Figure 4.5 illustrates the scenario in which the destination network is non-searchable, e.g. BitTorrent.

The rules for colors of lines in the Figures 4.4 and 4.5 are the same as the one in the Figure 4.3. The blank rectangles represent the format conversion happen in FOGP peer to convert the requests and responses between the formats defined by IFP protocol and by protocols of standard networks. Here the initiator can be a FOGP peer or a LOGP peer. The case in which a peer downloads files from P2P file-sharing network of which it is member is trivial, thus is not shown in the figures for the sake of clarity.

The destination network is searchable. The process of inter-network downloading illustrated in Figure 4.4 is sketched as following:

Step 1. The *initiator* peer, belonging to one P2P file sharing network, deliveries the search request to another network or a group of other networks or to all other networks, via OGP overlay using OGP unicast, OGP multicast or OGP broadcast respectively.

Step 2. The *recipient* peer which is a FOGP peer and belongs to destination network, upon receiving the search request, convert the search request from format of IFP to format defined by its network. It then launches the search on its network using search mechanism of the network via its Native node.

Step 3. Upon receive the search result, the FOGP peer sends the result and its contact information to the *initiator* node via OGP overlay.

Step 4. Upon receiving the search result from the recipient peer, if the result indicates the searched file exist on the destination network, the *initiator* peer directly contacts with the *recipient* asking it to retrieve certain file using IFP protocol.

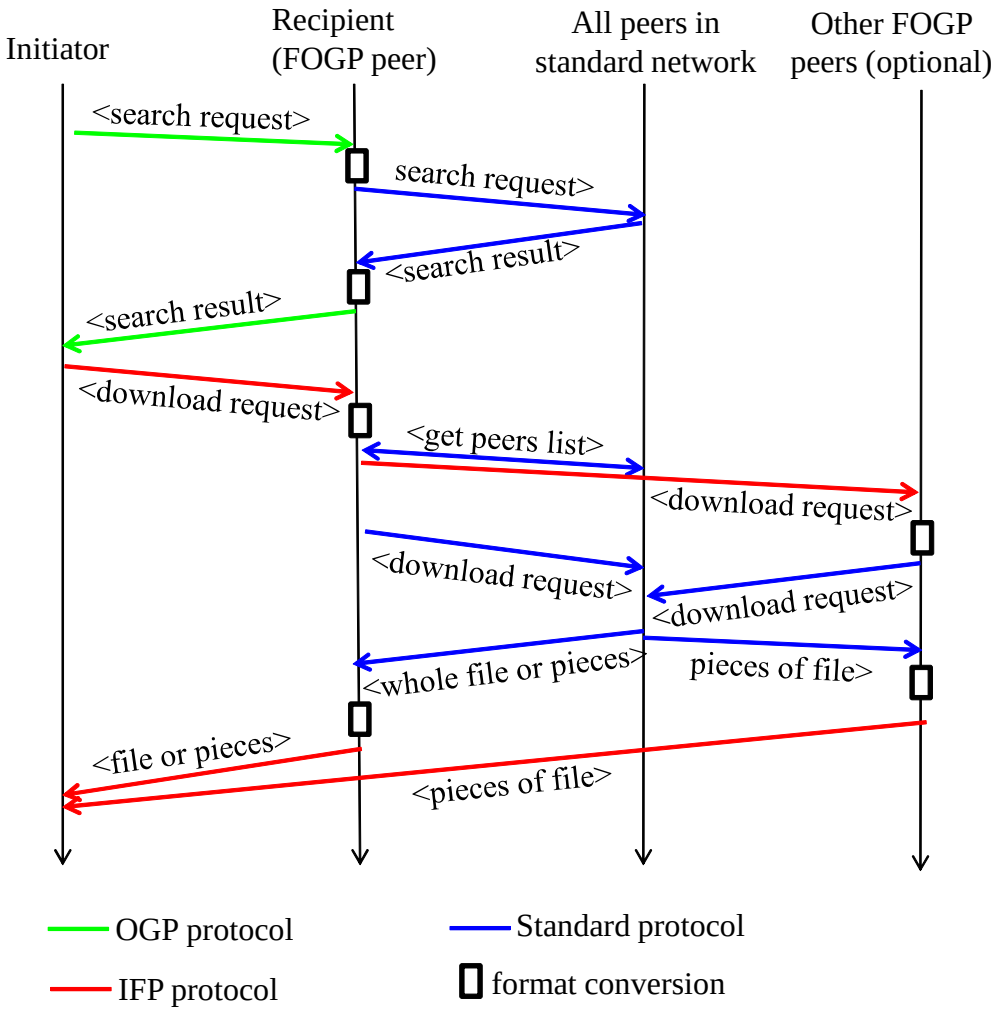

Figure 4.4: Inter-network download: the destination network is searchable

Step 5. The *recipient* peer, upon receiving the download request, retrieves the list of peers hosting the file via its Native node which obtain this list following the protocol of that network.

Step 6. If the destination network support multiple sources download, the recipient peer can, via IFP protocol, ask some other FOGP peers belonging to the destination network, which are in its FOGP peer list, to download some parts of the file. Otherwise, it is responsible for downloading the entire file.

Step 6. The file or the parts of file is then downloaded by these FOGP peers following the protocol of destination network.

Step 7. Upon receiving the file or the parts of file from hosting peers after issuing the download request, the FOGP peers including the *recipient* send the file or the file parts back to the *initiator* node following IFP protocol. In the later case, the information for joining the parts is sent along with these parts.

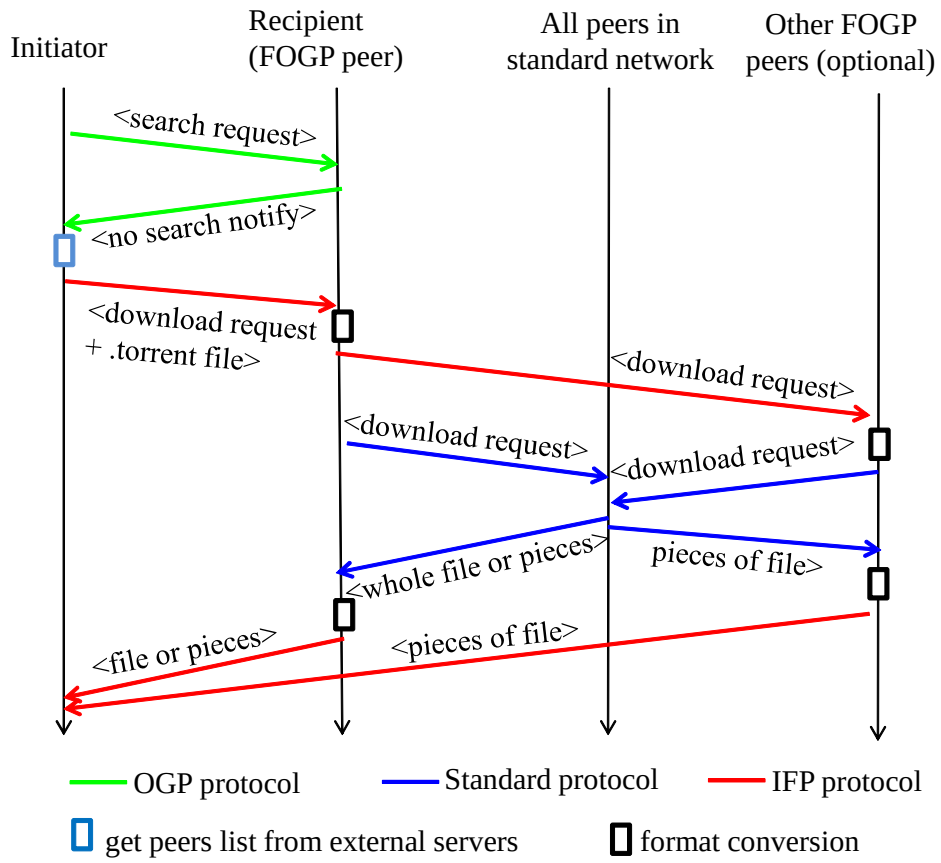

Figure 4.5: Inter-network download: the destination network is non-searchable.

The destination network is non-searchable. The inter-network downloading operation in the case in which the destination network is non-searchable (see Figure 4.5), has following differences comparing to the case the destination network is searchable.

- In step 2, upon receiving the search request from the *initiator* peer, the *recipient* notifies the initiator that its network doesn't support search capability instead of launches the search on its standard network.

- In step 3, upon receiving the notification, the *initiator* peer searches and retrieves the .torrent file from external server and then send the download request along with this information to the *recipient*.

From this point, the operations in two cases are the same.

4.3 IFP protocol

In a FOGP peer and a LOGP peer, the IFP protocol is run by *Cooperation Application*, therefore we use *Cooperation Application* with the meaning of IFP protocol in this section.

4.3.1 Tasks

The IFP protocol is responsible for the following tasks:

1. Launching the processes of searching, downloading and uploading files on P2P file-sharing network contains the peer and receiving the results via the Native node.
2. Launching the delivering of search requests or upload requests to P2P file-sharing networks don't contain the peer and receiving the results from these networks via the OGP node.
3. Introducing the intermediary formats of search requests, search results, download requests and upload requests and transcoding of those requests between the formats defined by P2P file-sharing networks and the intermediary formats.
4. Delivery of download requests on P2P file-sharing networks don't contain the peer and exchanging files.
5. Communicating with, and transferring the files between nodes including both FOGP and LOGP nodes.

4.3.2 Transcoding of messages

In our model, there is a set of messages are transferred back and forth between P2P file-sharing networks and OGP overlay. These messages belong to following categories: search request, search result, download request and upload request. Because the formats of these messages are various in different P2P file-sharing networks, the IFP protocol defines an intermediary format for each kind of message. As the messages cross back and forth between P2P file-sharing networks and OGP overlay, they are also transcoded between specific formats and intermediary formats respectively. The transcoding of these messages happens at the FOGP gateways.

Intermediary messages are defined by IFP protocol as following:

Search request. Most of P2P file-sharing networks have search capability with keyword search. The search criteria can include file attributes. One exception is BitTorrent, the most widely used P2P file-sharing network, which does not have search capability. However, BitTorrent users can still search the torrent files from websites using keywords. Therefore, IFP defines the intermediary search request containing keywords and file attributes

Search result. P2P file-sharing networks return the list of matched files for each search request. Hence, the intermediary search result contains the notification of no search capability in case of BitTorrent or the list of matched files along with attributes in other cases.

Download request. The intermediary download request contains the torrent file in case the destination network is BitTorrent and the information of the expected file in other cases.

Upload request. The intermediary upload request contains file attributes based on which the recipient FOGP decides to accept or not.

The four intermediary messages are illustrated in Figure 4.6.

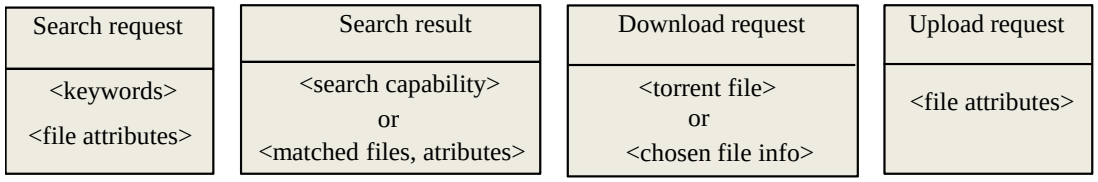

Figure 4.6: Formats of intermediary messages

4.3.3 Inter-network downloading

In this section, we describe the algorithm of IFP protocol for inter-network downloading scheme which allow user to download files from any P2P file-sharing network.

Algorithm description. The *Cooperation Application*, i.e. the IFP protocol, functions as follows:

Step 1. The *Cooperation Application* in the *initiator* peer asks the OGP node to send the search request to destination networks. The case that the *initiator* search files on its network is trivial, thus is not shown for the sake of clarity.

Step 2. The *Cooperation Application* in the *recipient* peer, which is a FOGP peer belonging to the destination network, upon receiving the search request, acts as follows: if the destination network is BitTorrent, it return BitTorrent indication i.e. no search capability via OGP node. Otherwise, the *Cooperation Application* converts the search request from IFP format to the format defined by the destination network. It then launches the search on this network via its Native node. Upon receiving the search result, the *Cooperation Application* converts this result to the IFP format and then sends the result along with the information of peer to the *initiator* via the OGP node.

Step 3. Upon receiving the search result from the *recipient* peer, if the result indicates the destination network as BitTorrent, then *the user* himself has to search and download the torrent file from a website. The *Cooperation Application* then send the torrent file to the *recipient* in the download request. Otherwise, if the destination network is not BitTorrent and if the sought file exists on the destination network, the *Cooperation Application* in *initiator* peer contacts with the *recipient* asking it to retrieve the file.

Step 4. The *Cooperation Application* in the *recipient* peer, upon receiving the download request, retrieves the list of peers hosting the file via the Native node. If the destination network supports multiple-source download, the *Cooperation Application* can ask some other FOGP peer belonging to destination network, which are in its FOGP peer list, to download some parts of the file. Otherwise, it is responsible for downloading the entire file using the Native node.

Step 5. The *Cooperation Application* in the *FOGP* peer, upon receiving the request for downloading some parts of the file, downloads these parts via the Native node.

Step 6. Upon receiving the file or file parts from the hosting peers after issuing the download request, the *Cooperation Application* in the recipients send the file or file

parts back to the *initiator* node and the information for joining the parts is sent along with these parts.

Pseudo code. In this paragraph, we present the pseudo code of above description.

First we introduce some notion used in the pseudo code.

- $CooApp, OGPNode, NativeNode$ are Cooperation Application, OGP node and Native node in a peer respectively.
- $Native\_Search\_Mess, IFP\_Search\_Mess$ are the search messages in the forms of native format and intermediary format respectively.
- $Native\_Search\_Result, IFP\_Search\_Result$ are search results in the forms of native format and intermediary format respectively.
- $IFP\_Download\_Mess, Download\_Part\_Request$ are messages for downloading a file and some part of a file defined by IFP protocol respectively.
- $contactInfo$ is the contact information of a peer.
- $destNets$ is destination P2P file-sharing networks which will receive the message.
- $part\_info$ is the information of some parts of a file.
- $torrent\_file$ is the .torrent file of a file which is got from *websites*.
- $send(), transcoding(), download()$ are methods for sending messages, transcoding of messages and downloading file or parts of the file respectively.

Algorithm 1 IFP protocol: inter-network downloading

Initiator: $CooApp.OGPNode.send(IFP\_Search\_Mess, destNets)$

on receipt of IFP_Search_Mess **from** $Initiator$
  if $(searchCapability == no)$
    $IFP\_Search\_Result = "BitTorrent"$
  else if $(searchCapability == yes)$
    $Native\_Search\_Mess = CooApp.transcoding(IFP\_Search\_Mess)$
    $Native\_Search\_Result = CooApp.NativeNode.search(Native\_Search\_Mess)$
  $IFP\_Search\_Result = CooApp.transcoding(Native\_Search\_Result)$
  $CooApp.OGPNode.send((IFP\_Search\_Result, contactInfo), Initator)$

on receipt of $(IFP\_Search\_Result, contactInfo)$ **from** $dest$
  if $(IFP\_Search\_Result == "BitTorrent")$
    get $torrent\_file$ **from** $websites$
    $CooApp.send(IFP\_Download\_Mess(torrent\_file), contactInfo)$
  else if $(IFP\_Search\_Result.contains(looking\_file\_info))$
    $CooApp.send(IFP\_Download\_Mess(looking\_file\_info), contactInfo)$

on receipt of $IFP\_Download\_Mess$ **from** $Initiator$
  if $(download\_from\_multiple\_source == true)$
    // ask M other FOGP peers, denoted by peer[i], belonging to the same standard
    // network to download parts of the file using part_info information
    $for\ (int\ i = 1; i < M; i++)$
      $CooApp.send(Download\_Part\_Request(part\_info, initiator\_Info), peer[i])$
    $file\_part = CooApp.NativeNode.download(part\_info)$
    $CooApp.send((file\_part, joining\_info), Initiator)$
  else
    $file = CooApp.NativeNode.download(file\_info)$
    $CooApp.send(file, Initiator)$

on receipt of $Download\_Part\_Request(part\_info, initiator\_Info)$
  $file\_part = CooApp.NativeNode.download(part\_info)$
  $AppNode.send(file\_part, joining\_info, Initiator)$

4.3.4 Inter-network uploading

The inter-network uploading scheme allows the users to upload their files to any network.

Algorithm description. The algorithm of IFP protocol, i.e. the *Cooperation Application*, in the inter-network uploading scheme is as follows:

Step 1. The *Cooperation Application* in the *initiator*, which is a FOGP or a LOGP peer, sends the upload request to *recipients* which are FOGP peers belonging to a group of networks that the *initiator* wants to replicate the file to, via its OGP or LOGP node.

Step 2. Upon receiving the request, the *Cooperation Application* in a *recipient* sends the response notifying the *initiator* whether the upload request is accepted or not via its OGP node.

Step 3. Upon receiving the notification, if the upload request is accepted, the *Cooperation Application* in the *initiator* peer sends the file to the *recipient*.

Step 4. Upon receiving the file, if the recipient's network is BitTorrent, the *Cooperation Application* in the *recipient* creates a torrent file for the file and registers the torrent file with some trackers, using its Native node. Then the *Cooperation Application* sends the torrent file back to the *initiator*. If the *recipient*'s network isn't BitTorrent, the *Cooperation Application* uploads the file to its standard network using the Native node and sends the acknowledgement back to the *initiator*.

Pseudo code. Beside the notion presented in the pseudo code of inter-network downloading algorithm, we introduce some more notion as following:

- $Upload\_Request, Acceptance\_Notify, Ack$ are messages defined by IFP protocol for requesting to upload a file, answering whether an upload request is accepted or denied and giving the acknowledgement respectively.
- $file\_attributes$ is attributes of a file.
- $upload(), createTorrent(), register()$ are methods for uploading a file, creating the .torrent file of a given file and registering a file with certain tracker respectively.

Algorithm 2 IFP protocol: inter-network uploading

Initiator: $CooApp.OGPNode.send(Upload\_Request(file\_attributes), destNets)$

on receipt of $Upload\_Request(file\_attributes)$ **from** $Initiator$
  $CooApp.OGPNode.send(Acceptance\_Notify, Initiator)$

on receipt of $Acceptance\_Notify$ **from** $dest$
  If $Acceptance\_Notify == "Accept"$
    $CooApp.send(file, dest)$

on receipt of $file$ **from** $initiator$
  if $(protocol! = BitTorrent)$
    $CooApp.NativeNode.upload(file)$
    $CooApp.send(Ack, Initiator)$
  else if $(protocol == BitTorrent)$
    $torrent\_file = CooApp.NativeNode.createTorrent(file)$
    $CooApp.NativeNode.register(torrent\_file)$
    $CooApp.AppNode.send(Ack(torrent\_file), Initiator)$

on receipt of $Ack(torrent\_file)$
  upload $torrent\_file$ **to** $websites$

4.4 Evaluation

4.4.1 Metrics

We characterize the cooperation efficiency of P2P file-sharing networks by two metrics: Rd and Ru. Here, Rd is the ratio of successful inter-network download operation. The Rd value shows the probability of the user successfully download a file which does not exist in user's network but exist in other networks. The second metric, Ru, is the ratio of successful upload operations. The Ru value shows the probability of the user successfully upload its file to networks which it is not a member.

In the cooperation model, FOGP peers in one network play the roles of gateways for other FOGP peers in that network and LOGP peers to exchange files with peers in other networks. We define some more metrics as following:

- The load on a FOGP peer is the number of files transferring through that peer on a period of time.
- The routing cost of search messages, i.e. the number of hops, on OGP overlay.
- The setting of a cooperating system is the percentage of FOGP peers and the percentage of LOGP peers in each network of that system.

4.4.2 Objectives and methodology

The evaluation section has three objectives. First is to identify the settings of evaluated systems which ensure the cooperation efficiency. Second objective is to investigate the

tradeoff between routing cost on OGP overlay and the load on each FOGP peer while the third is to investigate the scalability of our model.

To achieve the first objective, we evaluated the dependence of the cooperation efficiency on the percentage of FOGP and the percentage of LOGP by two steps. In the first step, we evaluated the dependence of the cooperation efficiency on the percentage of FOGP by varying the percentage of FOGP while setting the percentage of LOGP to 0. The result of this step is the percentages of FOGP peer which ensure the cooperation efficiency in the system without LOGP peers. In the second step, we fixed the percentage of FOGP peers to the values ensuring the cooperation efficiency obtained in the first step and evaluate the cooperation efficiency with the percentage of LOGP peers is varied. The results obtained from second step are the settings which ensure the cooperation efficiency of networks.

While the second objective is achieved by analyzing the routing cost of search messages and the load over FOGP peers, the third objective is obtained by investigate the cooperation efficiency under the increase of networks' sizes.

4.4.3 Setup

In this chapter, we evaluate the cooperation of three typical kinds of P2P file-sharing networks: (*i*) the network without search capability, (*ii*) the network with the flooding search and (*iii*) the network with DHT search. To achieve this, a complete system including implementations of three file-sharing networks based on BitTorrent, i.e. no search capability, Gnutella, i.e. flooding search and Kademlia, i.e. DHT search, are deployed on the French Grid5000 platform [G5k], which aims at providing a nation-wide test bed to study large scale parallel or distributed systems.

In each file-sharing network, a subset of peers is FOGP peers, another subset are LOGP peers and the rest are blind peers. Files that each of which is unique are randomly distributed across all of the networks. The FOGP and LOGP peers periodically perform two operations. First is downloading random files on any of the networks the peers don't belong to. Second is uploading their files to random networks they don't belong to.

All experiments are performed in high churn rate condition with the churn follows the Pareto distribution and the lifetime mean of nodes is set to 3600 seconds. Each experiment is run 5 times. Average values and corresponding relative standard deviations (*RSD*) of the metrics are plotted in the figures.

Experimental parameters	Scenario 1	Scenario 2	Scenario 3
% of FOGP peers	3, 5, 10, 20, 30	5, 10	6
% of LOGP peers	0	10, 20, 40, 60	0
No. of nodes per network	100		50, 100, 150, 200
Lifetime mean (second)	3600		
Type of networks	BitTorrent, Kad, Gnutella		

Table 4.1: Values of experimental parameters

4.4.4 Software

The software for evaluating the *cooperation solution* for cooperating P2P file-sharing networks constitute twofold:

JNode: a P2P node client, which implements: IFP, OGP, lightweight OGP protocols and three simple file-sharing networks based on three protocols namely Kademlia, Gnutella and BitTorrent respectively. It can be instantiated as OGP node, lightweight OGP node, Kademlia node, Gnutella node and Chord node. Every kind of node allows the users to search and receive files.

Manager: the experiment manager, which perform following tasks:

1. It automatically deploys the inter-connected file-sharing networks in large scale on Grid5000 platform using the configuration read from configuration file (a text file). In the configuration file, user can define various experiment parameters such as the number of overlays, the protocol of each overlay, the percentage of OGP, LOGP nodes, the duration time of the experiment, the time interval for lookup data, etc.

2. It launch the upload file, search and retrieve file processes of random nodes using the configuration read from configuration file.

3. It collects statistics of the experiments such as the success rate of upload file, download file operations.

The software is available on the website: http://www-sop.inria.fr/members/Giang.Ngo_Hoang/software.html

4.4.5 The cooperation efficiency on percentage of FOGP

In this investigation, each network contains 100 nodes including only blind peers and FOGP peers with the percentage of FOGP peer varies from 3% to 30%. The experimental parameters are given in Table 4.1, *Scenario 1*. The experimental results are illustrated in the Figure 4.7 in which the *Rd* and *Ru* values are represented by the blue line and the red line respectively.

From the figure, we can see that the *Rd* and *Ru* has same trend and *Ru* line stays above *Rd* line. The *Rd* comes from around 92 % to 97% when the percentage of FOGP peer rises from 3% to 5% and then slightly vary in the range from 97% to 99% as the percentage of FOGP peer increase from 5% to 30%. The *Ru* value increase from round 95% to 99% and then slightly vary in the range from 99% to 100% as the percentage of FOGP peer grow from 3% to 5% and then to 30%.

The Figure 4.7 also show that the *RSD* values of *Ru* and *Rd* metrics vary in the range from 0.6% to 2.7% as the percentage of FOGP peers vary from 3% to 30%.

Analysis and discussion. The experiments shows an interesting result that our model can achieve cooperation efficiency, namely *Rd* and Ru values are larger than 97% and 99% respectively, with only a small number of FOGP peer (not less than 5%). The low values of *RSD* show the stable of the experimental results.

The reason for the trend of *Rd* and *Ru* is as follows. Each file-sharing network has a number of FOGP peers serving as the gateways for exchanging message and data with other networks. In a high churn environment, some of them can go down for at certain time. During this time, if the number of FOGP peers in a network is too small, some FOGP peers in other networks do not have any backup gateways for the downed gateways of that network in their routing tables. With 3% of FOGP peer, each network has $3\% \cdot 100 = 3$ gateways and this number is not enough for the FOGP peers to build their routing table with adequate number of backup. While with 5% of FOGP peers,

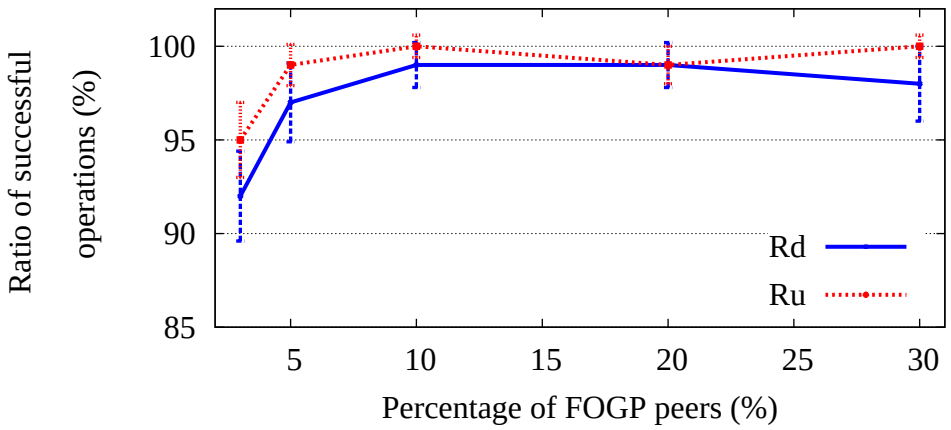

Figure 4.7: Cooperation efficiency vs. percentage of FOGP

there are 5 gateways per network, and this number appears to be sufficient for having quite enough backup. As the percentage of FOGP peers rises, values of slightly vary, since the routing table of the FOGP peers already has enough backup.

The inter-network download process constitutes more communication hops than inter-network upload process. In churn environment, each communication hop has a probability of failure due to the failure of peers; hence the success probability of inter-network download operation is smaller than the one of inter-network upload operation. That is why the *Ru* line stays a bit higher than *Rd* line.

4.4.6 The cooperation efficiency on percentage of LOGP

In this step, the percentage of FOGP peers is set to 5% and 10%, i.e. the percentage ensuring the cooperation efficiency in the system without LOGP, while the percentage of LOGP peer varies from 10% to 60%. The parameters of this scenario are given in Table 4.1, *Scenario 2*.

The experimental results are illustrated in the Figure 4.8. As the experimental results when the percentage of FOGP peer is set to 10% and 5% are similar, we only show the results in the case the percentage of FOGP peer is set to 10% on the figure for the sake of clarity. The Figure 4.8 shows that the *Rd* and *Ru* values slightly vary in the range from 98% to 99% and from 99% to 100% respectively when the percentage of LOGP peers varies in the ranges from 10% to 60%. In case the percentage of FOGP is set to 5%, the *Rd* and *Ru* values also slightly vary in the range from 97% to 99% and from 99% to 100% respectively as the percentage of LOGP peers stays in the ranges from 10% to 60%. The *RSD* values are less than 1.5% in all cases.

Analysis and discussion. We can see that our model achieves cooperation efficiency with every percentage of LOGP. This because the LOGP peers rely on FOGP peers for cooperation with peers in other networks while the communication between LOGP peers and FOGP peers is very efficient (this is shown in Chapter 3). The fact that our model achieves cooperation efficiency with every percentage of LOGP is an important result. It means that any user who has not enough resources to act as FOGP peer can act as LOGP peer with high cooperation efficiency.

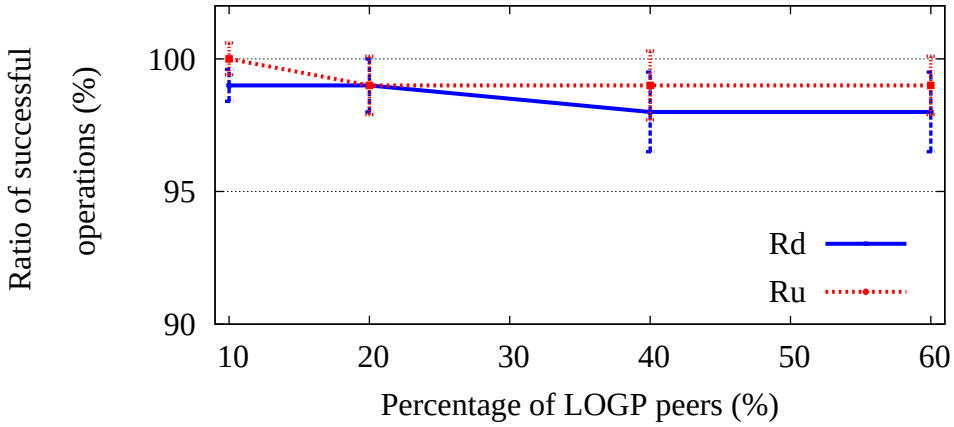

Figure 4.8: Cooperation efficiency vs. percentage of LOGP

The results of the two steps show that the settings ensuring the cooperation efficiency of our model are the ones in which the percentage of FOGP peers is not less than 5%. This simple condition leaves us a lot of flexibility in choosing the percentage of FOGP peers.

4.4.7 Tradeoff between routing cost and load on FOGP peers

In this section, we investigate the tradeoff between the routing cost of search messages on OGP overlay and the load on FOGP peers by *analysis*.

In our model, a peer can download a file which is in other network from one FOGP peer or multiple FOGP peers depending on the properties of the network containing the file. Each file or part of file is transferred though only one FOGP peer. Therefore, on average, we can assume if a file is downloaded, it is transferred though one FOGP peer. When a peer uploads a file, the file is also transferred via only one FOGP peer. We also assume that all FOGP peers and LOGP peers periodically download and upload files at the same rate. We define the unit of load is the number of files that a FOGP peer or a LOGP peer upload and download in a period of time.

The dependence of load on percentage of FOGP peers and percentage of LOGP peers and the dependence of routing cost on percentage of FOGP peers are shown in Figure 4.9 and Figure 4.10 respectively.

In Figure 4.9, the four lines with blue, red, green and brown colors illustrate the load on a FOGP peer in the cases the percentage of FOGP peers is set to 5%, 10%, 20% and 40% while the percentage of LOGP peers varies from 0% to 95%, 90%, 80% and 60% respectively. At 0% of LOGP peers, the load on each FOGP peer is one unit in all cases. As the percentage of LOGP peers increase while the percentage of FOGP peers is fixed, the load on FOGP peers increase to 20, 10, 5 and 2.5 units respectively due to the increase of number of files which are uploaded and downloaded by LOGP peers transferred through the FOGP peers.

In Figure 4.10, the blue line illustrates the routing cost over OGP overlay with the U parameter in the table 3.1 is set to 1 while the percentage of FOGP peer increases from 5% to 100% and the unit of y-axis is the routing cost over OGP overlay when the

percentage is set to 5%. As the percentage of peer increase from 5% to 100%, from the table 3.1, we can see that the routing cost over OGP overlay crease from 1 unit to around 5.3 unit as illustrated in Figure 4.10

From Figure 4.9 and Figure 4.10, we can see that when the percentage of FOGP peer increase while the percentage of LOGP peers is fixed, the load on each FOGP peer decreases because the load generated by LOGP peers over each FOGP peer decreases. On the other hand, when the percentage of FOGP peer decrease, the size of OGP overlay also decreases results in the routing cost of search messages over OGP overlay decreases. Therefore, when building a system of cooperating file-sharing networks, one need to tradeoff between the load on each FOGP peer and the routing cost of the search messages over OGP overlay, i.e. choosing a suitable percentage of FOGP peers. This tradeoff can be achieved by using Figure 4.9 and Figure 4.10.

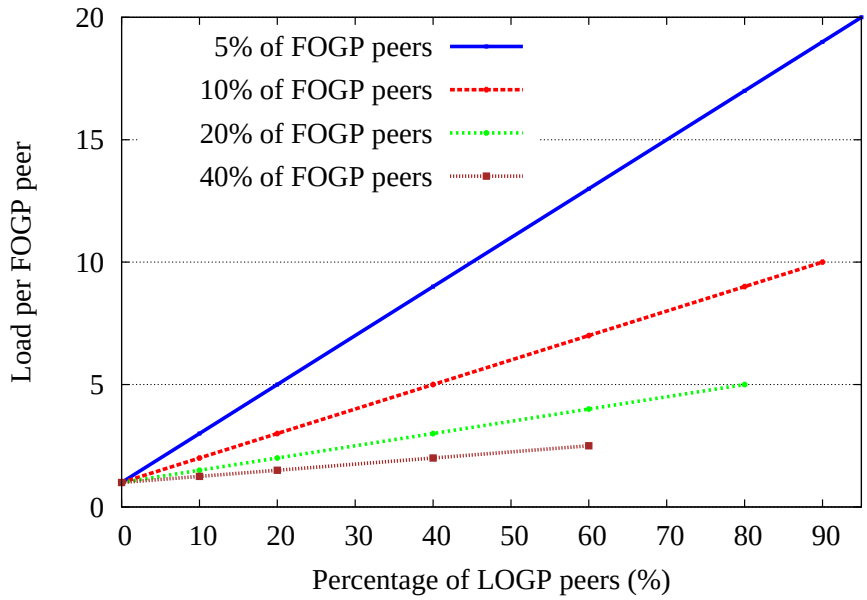

Figure 4.9: Load on a FCFS peer

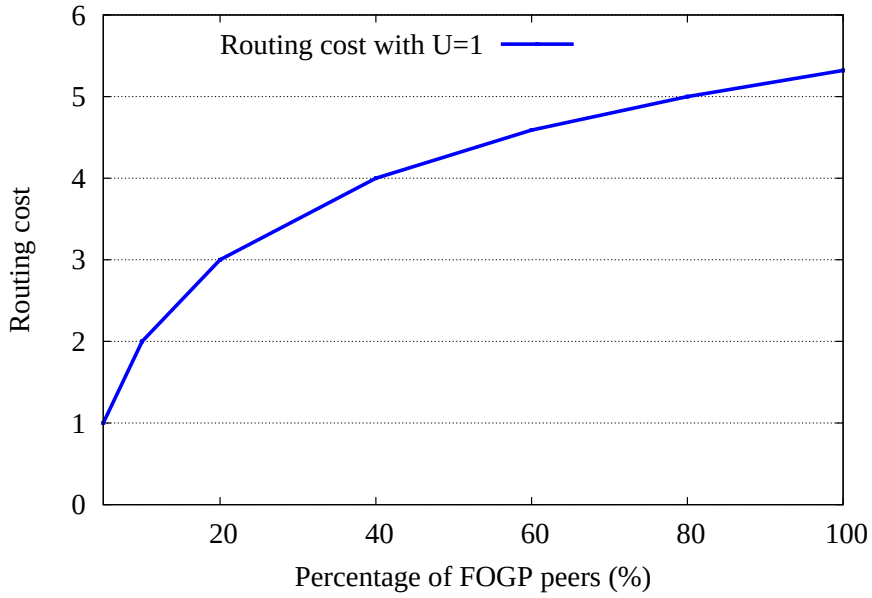

Figure 4.10: Routing cost of search messages

4.4.8 The scalability of the model

In this investigation, the percentage of the FOGP peers and the LOGP peers is set to 10% and 5% respectively while the number of nodes in each network increases from 50 to 200. The parameters of this scenario are given in Table 4.1, *Scenario 3*.

The experimental result are illustrated in the Figure 4.11. This figure shows that the *Rd* and *Ru* slightly vary in the range from 97% to 99% and from 99% to 100% respectively when the size of each file-sharing network increases from 50 nodes to 200 nodes.

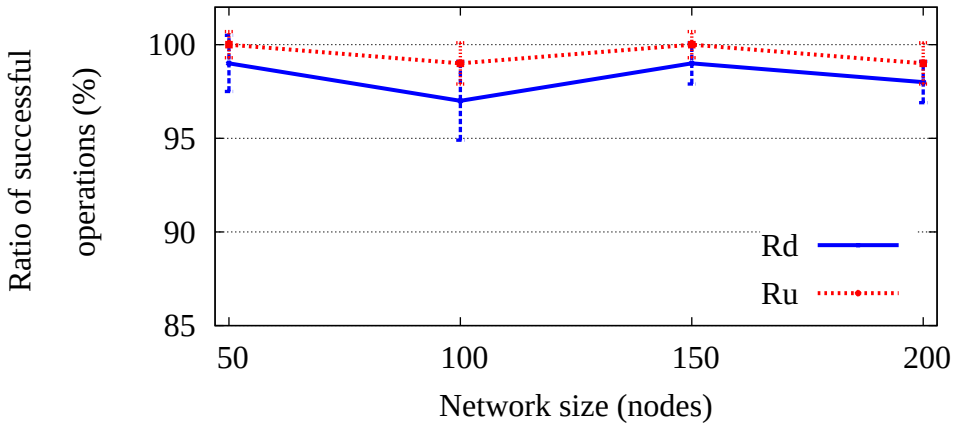

Figure 4.11: Cooperation efficiency vs. network sizes

Analysis and discussion. The fact that cooperation efficiency only slightly vary when the size of each network increase 4 times shows that our model is scalable with the scale

of participant networks.

4.5 Related work

As mention in the Chapter 3, there has been only one previous work [Konis at al 2006] on cooperation of P2P file-sharing networks. However, this study proposed the mechanism for cooperating only purely queries flooding-based file-sharing networks which constitute a very small portion of current file-sharing networks.

Author in [Konis at al 2006] evaluated their model by simulating two unstructured and purely query flooding-based file-sharing networks cooperating with each other. However, due to the large difference in application area, i.e. we evaluated our model with all three typical kinds of P2P file-sharing network today, therefore it is not adequate to compare the evaluation results between our work and the work in [Konis at al 2006]. Hence we did not perform the comparison here.

4.6 Chapter summary

In this chapter, we have introduced a model for backward compatible heterogeneous P2P file-sharing networks which can apply to all current P2P file-sharing networks. The model constitutes the IFP protocol, laying on OGP routing framework, for bridging P2P file-sharing networks at application level. This model serves as a case study of the framework for backward compatible cooperation of P2P systems presented in Chapter 3.

The evaluations show that having the percentage of FOGP peers not less than 5% with any percentage of LOGP peers is sufficient for achieving high cooperation efficiency in churn conditions: Rd and Ru are not less than 97% and 99% respectively as the lifetime mean of peer is set to one hour. This leaves us a great flexibility when choosing the percentage of FOGP peers. Our model also leaves us the ability to tradeoff between the load per FOGP peer and the routing cost over OGP overlay of search messages.

The experiments also show the scalability of our model with the Rd and Ru values are slightly vary in the ranges from 97% to 99% and from 99% to 100% respectively when the size of each network increase 4 times from 50 nodes to 200 nodes.

This chapter relies on following conference paper:

1. Giang Ngo Hoang, Luigi Liquori and Hung Nguyen Chan. Backward-Compatible Cooperation of Heterogeneous P2P Systems. The 15th International Conference on Distributed Computing and Networking, ICDCN 2014, accepted.

Chapter 5

A Scalable Communication Architecture for Advanced Metering Infrastructure in SmartGrid

Contents

5.1 Introduction

Cyber-Physical Systems (CPS) are "integrations of computation, networking, and physical processes in which embedded computers and networks monitor and control the physical processes, with feedback loops where physical processes affect computations and vice versa" [CPS]. Applications of CPS can be found in various areas. Smart grid, autonomous automotive systems, medical monitoring, process control systems, distributed robotics are some examples of CPS.

A common feature of many CPS systems is the large amount of sensors and actuators are spread over wide geographic areas for purposes of monitoring and controlling. One of the challenges for these systems is to scalable collect and process the huge amount of data generated by those sensors and actuators. In this chapter, we investigate the ability of using inter-connecting P2P overlays to address above challenge in smart grid, a typical example of CPS, as a case study.

Smart Grid is a data communication network integrated into the electrical grid that collects and analyzes data captured in near-real-time about power transmission, distribution, and consumption. Using the data, *Smart Grid* technology can provide predictive information and recommendations to utilities, their suppliers, and their customers on how to better manage the electric power [Cisco 2009].

According to [ami 2008], *Advanced Metering Infrastructure* (AMI) is a fundamental early step to grid modernization, i.e. Smart Grid. AMI is an integration of many technologies that provides an intelligent connection between consumers and system operators. In AMI system, smart meters measure and collect the energy consumption information, power quality from customers' premises. The metering data is, on-scheduled or on-demand, sent to *Metering Data Management System* (MDMS) which is a database with analytical tools allowing the interaction with system side applications such as *Consumer Information System* (CIS), *Outage Management System* (OMS), *Enterprise Resource Planning* (ERP). MDMS performs the validation, editing and estimation on the data and feeds the appropriate data to the system side applications to help optimize operations, economics and consumer service.

By current standards, a few kilobytes of data is collected from each smart meter every 15 minutes [Stand 2010, Bernaudo et al 2010]. In addition, metering data can be collected on demand for billing inquiries, outage extent verification, and verification of restoration [Khan et al 2013]. When the system is up to large scale, many existing communication architectures are not sufficient enough to deal with the waves of meter data due to the limitation in bandwidth.

In this chapter, we address above challenge by introducing a mix P2P and client-server communication architecture that allows scalable data collection, aggregation and management. The proposed communication architecture comprises multiple points of data collection and processing, i.e. MDMSs, which are geographically distributed and hierarchically organized. The metering data is collected, then is aggregated and transferred through multiple levels of MDMSs in a tree manner thus reduce the throughput of data after each level. While MDMSs are organized in P2P architecture to take the self-organization, scalability and resilience advantages of P2P, the connection from MDMSs to smart meters follows the client-server model as normally in order to be compatible with existing collectors and smart meters. As such, the main contribution of this chapter is the introduction of a new communication architecture for AMI featured by characteristics including scalability, resilience and partly self-organization, i.e. MDMSs network is self-organized.

Our chapter is organized as follows. Section 5.2 presents the background on AMI system. In Section 5.3, we introduce the related work and our focus. In Section 5.4, we describe the proposed AMI communication architecture and investigate the characteristics of the architecture. The communication infrastructure is evaluated by experiments on large scale platform Grid5000 [G5k] in Section 5.5. Finally, we draw conclusions and introduce the future work in the Section 5.6.

5.2 Background on AMI

AMI is a fully configured infrastructure to provide an essential link between the grid, consumers and their loads, and generation and storage resources [ami 2008]. Figure 5.1 illustrates the general architecture of AMI system.

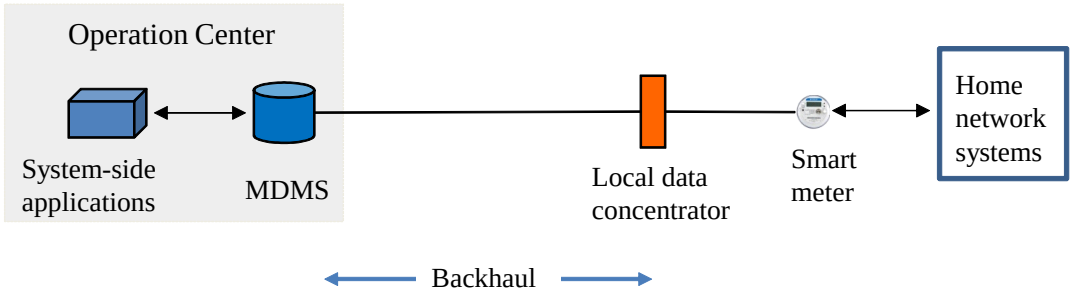

Figure 5.1: Overview of AMI

According to [ami 2008], the AMI system constitutes following components:

Home network systems which includes communicating thermostats and other in-home controls.

Smart meters which are programmable devices that perform various functions such as time-based pricing, consumption data for consumer and utility, loss of power (and restoration) notification, remote turn on / turn off operations, power quality monitoring, communications with other intelligent devices in the home, net metering, load limiting for demand response purposes, energy prepayment, tamper and energy theft detection, etc. Smart meters are connected to controllable electrical devices via home area network or local area network.

Communication networks from the smart meters to local data concentrators.

Local data concentrators collect metering data from smart meters.

Back-haul communications networks from concentrators to corporate data centers.

Meter Data Management Systems (MDMS) is a database with analytical tools allowing the interaction with system side applications such as Consumer Information System (CIS) which manages the utility billing and customer information, Outage Management System (OMS) which processes outage reports and display outage information to utility operators, power quality management and load forecasting systems, Geographic Information System (GIS) which captures, stores, manipulates, analyzes, manages, and presents all types of geographical data. One of the main tasks of AMI is to perform validation, editing and estimation on the AMI data to ensure that the data, fed to system side applications, is complete and accurate.

Data integration into existing and new software application platforms, i.e. system side application platforms.

5.3 Related work and our focus

Several communication architectures for AMI have been proposed in the last few years.

The traditional communication architecture [ami 2008] being one central *Operation Center* (OC) receiving metering data from all customers through local data concentrators. With this centralized architecture, all metering data go through MDMS at OC which then feeds the appropriate data into system side applications. This architecture makes the system simple and easy to manage. However, as the scale of system ups to large, this architecture is suffered from several non-scalable problems. First is the high possibility to create the data communication bottlenecks in the zones close to the OC, pointed out by Author of [Jiazh et al 2012]. Second is the unfeasibility in data processing due to the large data load, pointed out by Author of [Gerdes et al 2009].

The Author of [Jiazh et al 2012] introduced a model with some similarities with *Content Delivery Network* (CDN) [CDN] in which a central MDMS connects to multiple distributed MDMSs. They introduced an algorithm to calculate the optimal deployment of distributed MDMSs in their model. The distributed architecture allows the aggregation of data thus solves the problem of non scalability in data collection.

The Author of [Meili et al 2013] proposed an infrastructure based on group communication using hybrid adaptive multicast over public networks. By relying on group communication over public networks, this approach reduces the cost of investment but the reliability of communication as well as the latency of data collection are not guaranteed.

The Author of [Arena et al 2010] investigate the storage and monthly billing processing architecture. They compared storage techniques including the centralized relational database, the distributed relational database and the key-value distributed database storage. Their work focuses on achieving the scalability in processing metering data rather than achieving the scalability in collecting the metering data.

The Author of [Rusitschka et al 2010] proposed a model of using cloud computing for smart grid data management. The advantages of computing are the ability to provide huge storage, powerful processing and communication. In their work, while focusing on providing sufficient resources for smart grid data management, they do not investigate the distribution of necessary resources, i.e. points of data collection and management.

In our work, we focus on introducing a communication architecture for scalable data collection and management. Author of [Jiazh et al 2012] also proposed a communication architecture for scalable data collection using a CDN like model for MDMSs network. In our work, the MDMSs are connected by a P2P network which is more resilient with failure than the model proposed in [Jiazh et al 2012]. The scope of our work does not cover the investigation of data processing and integration algorithms.

5.4 AMI architecture

DHTs are scalable P2P architectures. However, their flat structure is not suitable for the aggregation of metering data. Hierarchy is probably the most natural structure for this purpose. Many tree-based overlays have been introduced such as [Jagadish et al 2005, Li et al 2006, Leitao et al 2007]. These overlays are designed for supporting systems with a large number of peers whose roles are equal, i.e. a peer can stay at any position in the tree. Under the failure of certain peers, other peers are reorganized and can stay at any level in the tree. However, in AMI system, functions of MDMSs at different levels are not equal. The MDMSs at highest level, i.e. closest to smart meters, process raw data and send the summarized data to MDMSs at lower level and so on. Therefore, we introduce

a new hierarchical architecture for AMI in which the levels of MDMSs keeps remain after reorganizing. In this architecture, DHTs are used to connect groups of MDMSs at the same level to take their self-organization and deterministic routing advantages.

5.4.1 Structure

Figure 5.2 illustrates the proposed AMI communication architecture. The AMI comprises multiple OCs which are geographically distributed and hierarchically organized. Each OC contains a MDMS for collecting and analyzing metering data from customers belonging to the area that the OC is responsible for. The MDMSs are equipped with analytical tools to interact with system-side applications which locate in that OC. As each OC has one MDMS, we use a cube to present both an OC and the perspective MDMS. Applications in an OC are presented by a can.

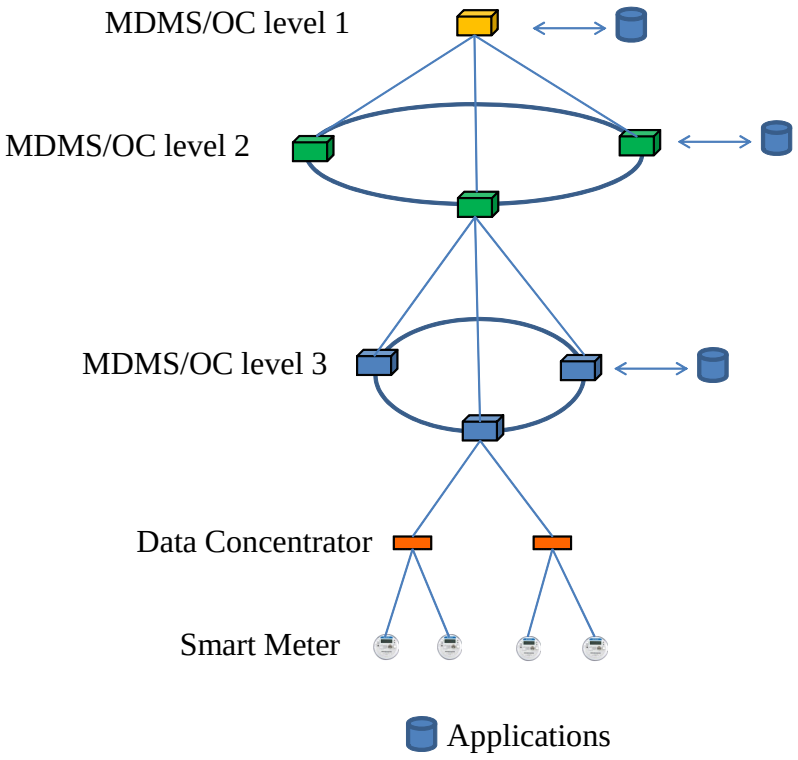

Figure 5.2: AMI proposed architecture

The level-1 MDMS is responsible for whole area where AMI covers. The whole area is divided into multiple sub-areas that each of which is covered by a level-2 MDMS. An area covered by a level-2 MDMS can be further divided into multiple sub-areas that each of which is covered by a level-3 MDMS and so on.

The level-n MDMS managing area S_n and the level-$(n+1)$ MDMS managing area S_{n+1} such that $S_{n+1} \subset S_n$ are called parent and child of each other respectively.

All MDMSs are connected in P2P architecture as following:

- All children of a level-n MDMS with $n \geq 1$ are organized in a DHT overlay network. The MDMS at level-n keeps contacts of all of its MDMSs children at level-$(n+1)$

while a MDMS at level-$(n+1)$ maintains the contact of its parent and contacts of parent's neighbors in the parent's overlay. In case that the parent MDMS is at level-1, its children maintain the contact of parent only.

- When the parent of a MDMS fails, the MDMS replaces its parent by the closest neighbor of the parent. If the old parent recovers from failure, the MDMS automatically changes the parent back to the old one. A MDMS also periodically check the aliveness of its children and remove the failed children from its children list.

Each MDMS at highest level manages a number of data concentrators following the client-server model. Each of the data concentrator collects metering data from a number of smart meters.

5.4.2 Data collection and processing

The processes of collecting and processing data is illustrated in the Figure 5.3. The information can travel in the system in three directions: upward direction, downward direction and intra-overlay.

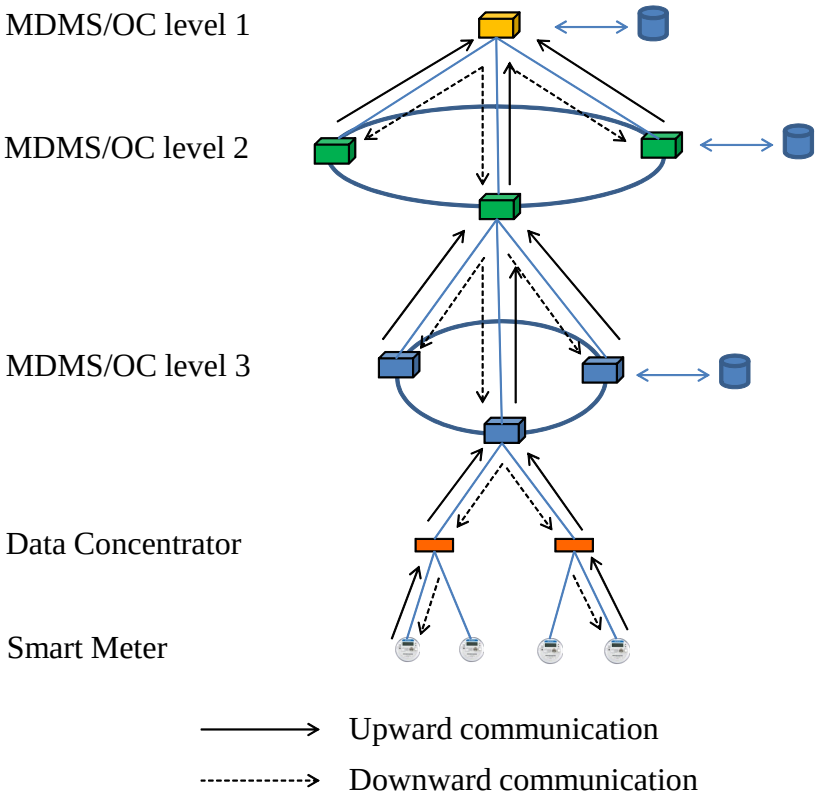

Figure 5.3: AMI proposed architecture

Upward direction. The upward communication is illustrated by solid arrows in the Figure 5.3. In this direction, customers on-scheduled or on-demand send metering data to local concentrators which then send the data to the MDMSs managing them. At each MDMS, the data is processed and aggregated and the appropriate

data is fed to local system-side applications, only the summarized data is sent to parent MDMS and so on.

Downward direction. The downward communication is illustrated by dashed arrows in the Figure 5.3. In this direction, the MDMS at level-n can send control commands or data to MDMSs at level-$(n+1)$. The data and commands such as new energy price, request for reducing the load can also be send to customers through concentrators.

There are three schemes of downward communication including broadcast, multicast or unicast. A MDMS at level-n can send the data or command to all of its children MDMSs or to a group of its children MDMSs or to one of its children MDMSs at level-$(n+1)$. A MDMS at highest level can send the data or command to all of its users or to a group of its users or even to a specific user.

Intra-overlay. In this direction, the data and commands or other information can be exchanged between MDMSs in one overlay. The communication follows the protocol of that overlay.

System-side applications are deployed in OCs at appropriate levels depend on their functions, characteristics and requirements on latency. The applications that is sensitive with latency such as OMS, *Demand Response* (DR) should be deployed in OC at highest level, i.e. close to the smart meters, while other applications which are less sensitive with latency such as Billing system can be located in OC at lower level.

The distribution of applications in high level OCs also help to reduce the throughput of data flowing toward the lower level OCs thus make the system more scalable in term of data communication.

The MDMS and system-side applications co-located in one OC can interact with each other in several patterns. The MDMS can on-demand or on-scheduled feeds data to system-side applications. Other way of interaction exploits the Publish-Subscribe model [PubSub] in which these applications subscribe for certain kind of event such as the outage flags generated by AMI system so that these applications are notified when the events happen. To enable these interactions, the utility can deployed analytical tools or publish-subscribed applications a long with the MDMS to provide the appropriate data to the applications.

We note that the information from system-side applications can be sent to customers via other paths such as Internet instead of AMI infrastructure. However, these communication paths are outside the scope of this work.

5.4.3 System analysis

This section investigates the characteristics of our architecture. We first define the meaning of some notation used in this chapter. The $\stackrel{\Delta}{=}$ notation means "equal" while the $<<$ notation means "much less than". The parameters of the model are denoted as following:

- M: the number of smart meters in the system;
- L: the largest level of OCs;
- λ_i: the number of messages generated by a smart meter in a period of time;

- α_n: the load over a MDMS at level n, calculated by a number of messages go to this MDMS. The load over a MDMS shows two aspects: the throughput of data flowing to the OC and the power processing needed to process these messages.

- β_n: the number of messages that a MDMS at level $n+1$ sent to its parent MDMS at level n in a period of time;

- $z_n \triangleq \frac{\beta_{n+1}}{\beta_n}$ with $1 \leq n \leq L-1$.

In case n=L, we let z_L be the number of messages a data concentrator sent to MDMS managing it over the number of messages that it receives from smart meters.

- K_n: the number of MDMSs in all OCs at level n;

We note that $z_n << 1$ because MDMS only sent summary data to its parent. This results in $K_n << K_{n-1}$.

Load scalability. According to [Scala], load scalability is the ability for a distributed system to easily expand and contract its resource pool to accommodate heavier or lighter loads or number of inputs. We define the load on AMI is the number of messages generated by all smart meters in one period of time which is calculated as following:

$$\sum_{i=1}^{M} \lambda_i \triangleq M \cdot \overline{\lambda}$$

with $\overline{\lambda}$ is the average traffic generated by a smart meter in a period of time. Thus the number of messages received by operation centers at level-n is:

$$M \cdot \overline{\lambda} \cdot \prod_{j=L}^{n} z_j$$

hence the load over a MDMS in level n is

$$\alpha_n \triangleq \frac{M \cdot \overline{\lambda} \cdot \prod_{j=L}^{n} z_j}{K_n} \tag{5.1}$$

with $n = \overline{1, L}$.

We note that in centralized model, the load on the central MDMS, denoted by α is:

$$\alpha \triangleq M \cdot \overline{\lambda} \tag{5.2}$$

From Formula 5.1 and Formula 5.2, we can see following points:

- By distributing the MDMSs and aggregating data at multiple levels, our model reduces the load on each OC comparing to the centralized model.

- By turning the number of aggregation levels and the number of OCs at each level, i.e. turning L and K_n, utility can control the load on each OC, i.e. control throughput and power processing requirement for each OC, thus eliminating the data transmission congestion as well as the overload in power processing.

  The adding or removing MDMSs to or from MDMS overlays can be achieved easily thanks to the self-organization advantages of P2P. If the change of load happens in a local area, utility only need to adapt the number of MDMSs in the OCs which are responsible for that area.

  The easiness of changing the resource pool to adapt with either the global changes or the local changes of load shows the load scalability of AMI based on our architecture.

Failure resilience. The proposed communication architecture exploits the resilience with failure advantage of P2P in managing the MDMS network.

According to [Cascio], resilience means the capacity of a system to withstand sudden, unexpected failures, and (ideally) to be capable of recovering quickly afterwards. Resilience implies both strength and flexibility in the sense that a resilient structure would bend, but would be hard to break.

In our proposed architecture, the data of a MDMS can be replicated at its neighbors in the DHT overlay. As the MDMS fails, the closest neighbor automatically replaces it. On the other hand, a MDMS normally sends the data to its parent MDMS. In case the parent MDMS fails, the MDMS automatically send the data to the closest neighbor of the parent MDMS.

Similarly, a data concentrator normally sends the data to its parent. In case the parent fails, it will send the data to closest neighbor of the parent. To deal with the situation that the level-1 OC fails, utility can deploy P2P applications for retrieving, querying data over overlay of MDMS at level-2.

As a matter of fact, even if the failure happen in some MDMSs, the system will quickly recovers and then works normally.

Low cost of maintenance and operation. With the large scale and the growth of AMI infrastructure, the self-organization ability of the MDMS network is very important in the sense that it helps the utility reduces the operation and maintenance cost. When the failures happen, the network automatically recovers without the human intervention. As the scale of system change, utility can easily add or remove MDMSs with little early configuration.

5.5 Evaluation of communication infrastructure

In this section, we evaluate the communication efficiency, the maintenance traffic and the scalability of the proposed communication architecture.

5.5.1 Implementation of communication architecture

We implemented and evaluated the fundamental part of the communication architecture in which one MDMS overlay at level-n connects to multiple MDMS overlays at level-

$(n+1)$. We extended the Chord [Stoic et al 2001], a typical DHT, to implement these MDMS overlays as a case study. Other DHTs can also be used instead.

The other parts, namely the MDMS level-1 connect to overlay of level-2 MDMS and a MDMS connects to data concentrators in client-server architecture, are well investigated architectures. Therefore we did not implement and evaluate these parts.

The implemented architecture is illustrated in Figure 5.4.

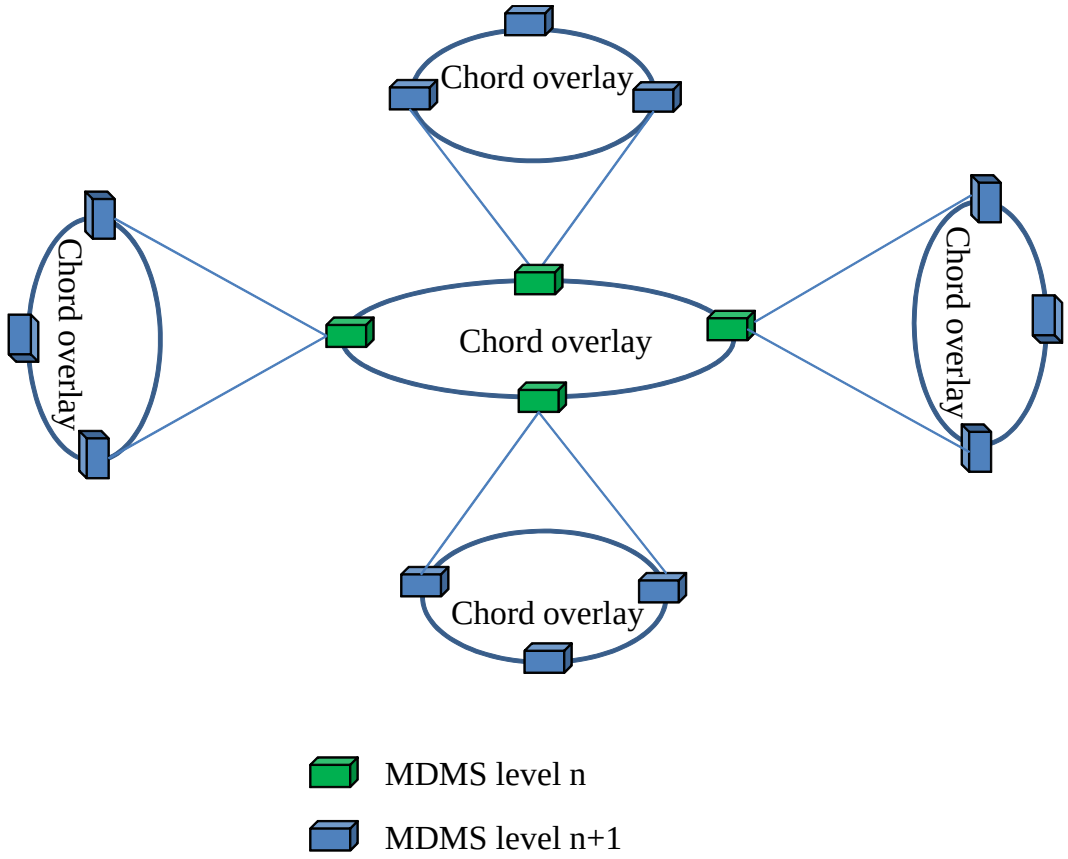

Figure 5.4: Evaluation communication architecture

5.5.1.1 ID assignment.

We assign an unique p-bits identifier to each MDMS as following:

- The level-1 MDMS is assigned a random p-bits number.
- Each of the level-2 MDMSs is assigned a p-bits number in which q first bits is random and $p-q$ last bit is set to 0.
- Assume that a level-n MDMS with $n \geq 2$ has identifier with first $(n-1) \cdot q$ bits is m_0. Its MDMSs children, i.e. the MDMSs in OC at level-$(n+1)$ has identifiers specified as following: first $(n-1) \cdot q$ bits is m_0, next q bits is random and $p - n \cdot q$ last bits is set to 0.

Figure 5.5 illustrates a simple example of ID assignment. The inter-connecting DHT structure includes three levels (see Figure 5.3) in which one level-1 MDMS connects to an overlay network of level-2 MDMSs. Each of the level-2 MDMS connects to an overlay network of level-3 MDMSs. For the sake of simplicity, in the example, the identifiers of MDMSs are presented by only 4 bits numbers, i.e. $p = 4$, here we set $q = 2$.

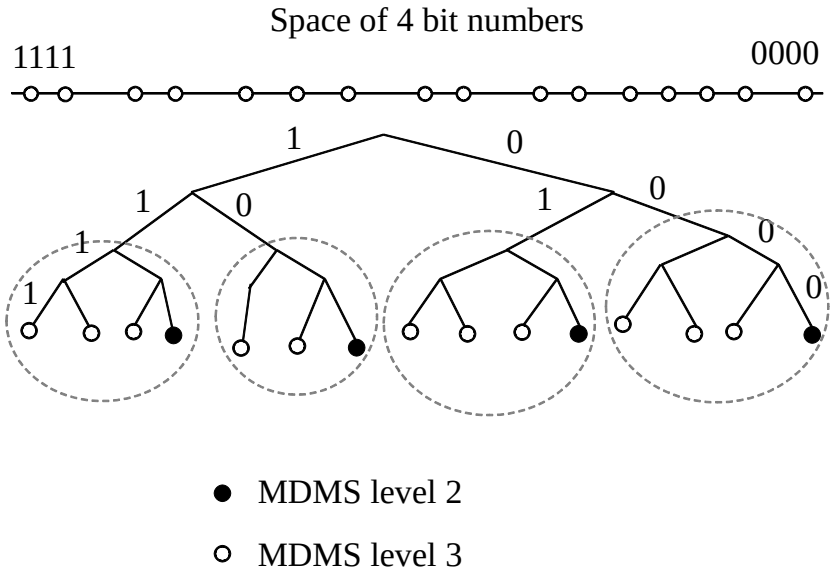

Figure 5.5: ID assignment example

In the Figure 5.5, the identifier of level-1 MDMS is arbitrary. The identifiers of level-2 MDMSs are presented by black circles with two last bit is set to 0. Each of the level-2 MDMS has several children, i.e. level-3 MDMSs, whose identifiers are represented by plain circles and share the first two bits with the level-2 MDMS. Therefore, a level-2 MDMS and its children belong to the same subtree bounded by a dashed circle.

5.5.1.2 Parent-children relationship

Parent-children assignments. A MDMS `X` in the child overlay is the child of a MDMS `Y` in the parent overlay if identifier of `X` stays between identifier of `Y` and identifier of `Y`'s successor, i.e. the closest succeeding node of `Y` in the identifier space.

Let `X@Y` denotes `X` is child of `Y`. Assume that `Z` is successor of Y. $\texttt{ID}_\texttt{X}$, $\texttt{ID}_\texttt{Y}$ and $\texttt{ID}_\texttt{Z}$ are identifiers of `X`, `Y` and `Z` respectively. Then we have that:

$$\texttt{X@Y} \textit{ if } \texttt{ID}_\texttt{X} \in [\texttt{ID}_\texttt{Y}, \texttt{ID}_\texttt{Z})$$

Figure 5.6 illustrates an example in which the parent MDMS overlay, i.e. O1, connects to three children MDMS overlays, namely O2, O3 and O4. In the figure, MDMSs are denoted by black circles. A, B and C are MDMSs in O1. C is successor of B which is successor of A. D, E and F are MDMSs in O2 and are children of A because $\texttt{ID}_\texttt{D}$, $\texttt{ID}_\texttt{E}$ and $\texttt{ID}_\texttt{F}$ belong to $[\texttt{ID}_\texttt{A}, \texttt{ID}_\texttt{B})$. Similarly, G, H and I are MDMSs in O3 and are children of C while K, M and N are MDMSs in O4 and are children of B.

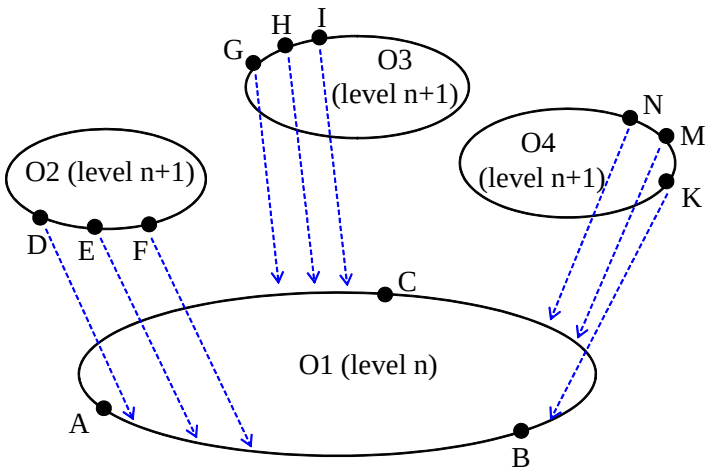

Figure 5.6: Parent children example

Children list and parent list. A peer maintains a children list and a parent list to keep the information of its children and parents. The children list of a peer contains information about all of its children. The parent list contains the information about the parent peer and parent peer's predecessor peers, i.e. the closest preceding peers of the peer in the Chord identifier space.

5.5.1.3 Communication schemes

Utility can employ various communication schemes to support communication among MDMSs. These schemes can be categorized into three categories based on the direction of the communication: intra-overlay communication, downward communication and upward communication.

Intra-overlay communication is the communication between MDMSs in one DHT overlay. A MDMS can employs unicast, broadcast over DHT and multicast over DHT for intra-overlay communication. In this chapter, we called these schemes intra-unicast, intra-multicast and intra-broadcast to distinguish them with similar schemes in downward communication.

Many studies such as [El-Ans et al 2003, Castro et al 2006] proposed algorithms for broadcast over a DHT. In our implementation, we used the broadcast algorithm proposed by [El-Ans et al 2003] as intra-broadcast while the intra-multicast algorithm is adopted from [El-Ans et al 2003] by ourselves.

Downward communication is the communication from parent MDMSs to children MDMSs. A MDMS can use any of the three schemes: downward-unicast, downward-multicast and downward-broadcast to respectively send information, i.e. data or commands, to its specific child MDMS or to a group of its children MDMSs or to all of its children MDMSs. For example, in the Figure 5.6, MDMS A can send information to a specific MDMS among D, E and F or to a group of or to all of them.

Upward communication is the communication from a MDMS to its parent MDMS. A MDMS use upward-unicast to send information to its parent MDMS. For example, in the Figure 5.6, MDMSs K, M and N can send information to B which is their parent MDMS.

Combination scheme. The combination between intra-overlay communication and downward communication allows the utility to unicast, multicast or broadcast information, i.e. data or command, to a specific MDMS, a group of MDMSs or to all MDMSs in any area.

The idea of combination scheme is as following. The sending MDMS first employs intra-unicast or intra-multicast or intra-broadcast to send the information to a specific MDMS or to a group of MDMSs or to all MDMSs in the same overlay with it. The receiving MDMS then forwards the information to one of its children or to group of its children or to all of its children.

The combination scheme constitutes three kind of communication as following:

- Combination-unicast: used by a MDMS at level-n to send information of a specific MDMS at level-$(n+1)$. For example, in the Figure 5.6, MDMS A can send information to a specific MDMS belonging to any of three overlays: O2, O3 and O4.
- Combination-multicast: used by a MDMS at level-n to send information of a group of MDMS at level-$(n+1)$. For example, in the Figure 5.6, MDMS A can send information to a group of MDMSs belonging to any of three overlays: O2, O3 and O4.
- Combination-multicast: used by a MDMS at level-n to send information of all MDMSs at level-$(n+1)$. For example, in the Figure 5.6, MDMS A can send information to all of the MDMSs in three overlays: O2, O3 and O4.

5.5.2 Evaluation

5.5.2.1 Objectives

This section evaluates following characteristics of the proposed P2P architecture: (i)The efficiency of communication in term of the ratio of successful communication; (ii) The traffic for maintaining the network of MDMSs characterized by the amount of traffic generated by a MDMS in a period of time; (iii) The scalability of the communication architecture.

The intra-overlay communication was evaluated in many previous studies such as [Stoic et al 2001, El-Ans et al 2003]. Therefore we focus on evaluating the efficiency of combination scheme and upward communication including: combination-unicast, combination-multicast and upward-unicast.

In this work, the scalability of the communication architecture is evaluated through the change of communication efficiency and the change of communication cost, i.e. generated traffic, as the scale of the system increase. In the experiments, the scale of the P2P system increased up to 4 times: the number of MDMSs in parent overlay increased from 10 to 40 while the number of children overlays increases from 10 to 40, i.e. the number of children MDMSs increased from 500 to 2000.

5.5.2.2 Experiment setup

Evaluation architecture. The evaluation architecture is illustrated in the Figure 5.4. In the experiment, a P2P system with one level-2 MDMS overlay connects to various numbers of MDMS overlays at level-3 has been deployed on the large scale French Grid5000 platform [G5k].

Evaluation scenario. Assume that a data concentrator receive the metering data from 100 smart meters while each MDMS receive the data from 100 data concentrators on average. Thus a MDMS manages $100 \cdot 100 = 10000$ smart meters on average.

We evaluate the communication architecture for large scale AMI with the number of smart meter from 5 millions to 20 millions. In this scenario, the number of MDMS at highest level (level-3) varies from $5000000/10000 = 500$ to $20000000/10000 = 2000$.

Assume that each MDMS level-2 manages 50 MDMSs level-3, thus the number of MDMS at level-2 varies from $500/50 = 10$ to $2000/50 = 40$. This also means that the number of children overlay varies from 10 to 40.

The experiments are performed in both no churn and high churn environments to show the efficiency of communication architecture in ideal condition and the resilience of the communication architecture under the failure of MDMSs respectively. In churn condition, lifetime mean of parent peers is set to 2 hours while lifetime mean of children peers is set to 1 hours.

Each experiment is run 5 times. The average values and standard deviation of evaluated metrics are plotted in the figures in following sections.

The values of experiment parameters are described in the Table 5.1.

No. of parent overlay	1
No. of children overlay	10, 20, 30, 40
No. of node per child overlay	50
No. of parent nodes	10, 20, 30, 40
Churn	high churn, no churn

Table 5.1: Values of experiment parameters

5.5.2.3 Software

The software for evaluating the AMI communication architecture constitute twofold:

Node: a P2P node client, which can be instantiated as child node or parent node.

Manage: the experiment manager, which performs following tasks:

1. It automatically deploys the inter-connected DHTs in large scale on Grid5000 platform using the configuration read from configuration file (a text file). In the configuration file, user can define various experiment parameters such as the number of children overlays, the number of nodes per child overlay, the duration time of the experiment, etc.

2. It launches the different communication schemes between children peers and parent peers.

3. It collects statistics of the experiments such as the success rate of communication and traffic generated by nodes.

The software are available on the website: http://www-sop.inria.fr/members/Giang.Ngo_Hoang/software.html

5.5.2.4 Experiment results: Communication efficiency and Scalability

This section shows the experimental results on the ratio of successful communication in combination scheme and upward communications.

Combination-multicast. The ratio of successful communication performed in combination-multicast scheme is illustrated in the Figure 5.7.

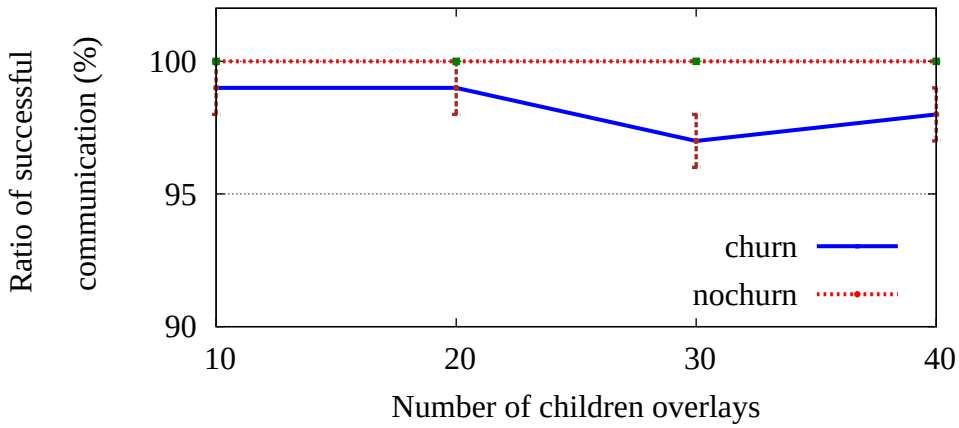

Figure 5.7: Success ratio in downward-multicast.

The lines "churn" and "nochurn" represent ratio of successful communication, performed in the systems under churn condition and no churn condition respectively, when the number of children overlays increases from 10 to 40.

The Figure 5.7 shows that the combination-multicast communication performed in the no churn environment succeed with the ratio of 100%. In the churn environment, the value of this ratio slightly changes in the range from 99% to 97% as the number of children overlay increases from 10 to 40 and the standard deviations are less than 1.5 for all cases. Assume that $\Delta_{com\_multicast\_churn}$ and $\Delta_{com\_multicast\_nochurn}$ are the percentage change of successful communication ratio in combination-multicast as the scale of the system increases in the churn and no churn environment respectively. Then:

$$\Delta_{com\_multicast\_nochurn} = \frac{100 - 100}{100} = 0\%$$

and

$$\Delta_{com\_multicast\_churn} = \frac{99 - 97}{99} = 2\%$$

Combination-unicast and upward-unicast. The Figure 5.8 illustrates the ratio of successful communication in combination-unicast and upward-unicast schemes. The two lines "combination-unicast" and "upward-unicast" represent the ratio of successful combination-unicast communication and upward-unicast communication respectively, performed in the systems which are under high churn condition, as the number of children overlay rises from 10 to 40.

From the Figure 5.8 we can see that the line "combination-unicast" is slightly changes in the range from 97% to 99% with the increase of number of children overlays from 10 to 40. The standard deviations are less than 1.5 for all cases. On the other hand, the line "upward-unicast" is horizontal line at value of 1. We did not show the ratio of successful communication in combination-unicast and upward-unicast in no churn condition which are 1 for all cases for the sake of clarity.

Assume that $\Delta_{com\_unicast\_churn}$ and $\Delta_{com\_unicast\_nochurn}$ are the percentage change of successful communication ratio in combination-unicast as the scale of the system increases in the churn and no churn environment respectively. Then:

$$\Delta_{com\_unicast\_nochurn} = \frac{100 - 100}{100} = 0\%$$

and

$$\Delta_{com\_unicast\_churn} = \frac{99 - 97}{99} = 2\%$$

Assume that $\Delta_{up\_unicast\_churn}$ and $\Delta_{up\_unicast\_nochurn}$ are the percentage change of successful communication ratio in upward-unicast as the scale of the system increases in the churn and no churn environment respectively. Then:

$$\Delta_{up\_unicast\_churn} = \Delta_{up\_unicast\_nochurn} = \frac{100 - 100}{100} = 0\%$$

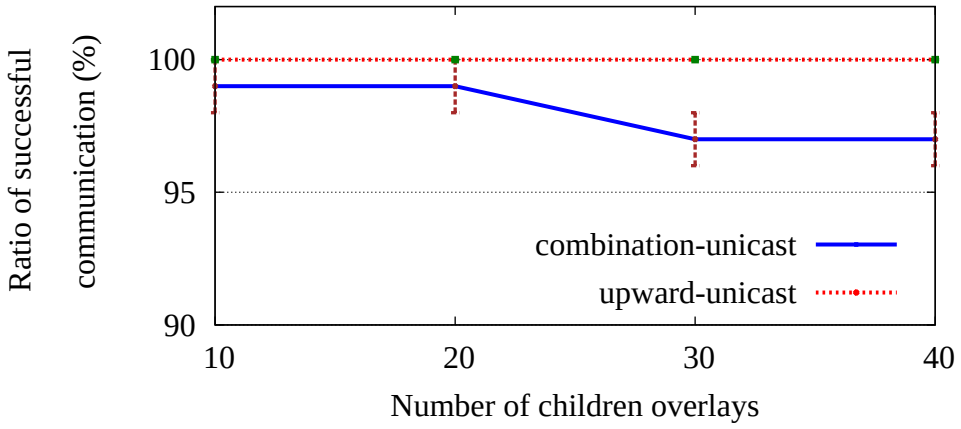

Figure 5.8: Ratio of successful communication in unicast schemes.

The percentage changes in communication efficiency in various communication schemes as the scale of the system increase up to 4 times is summarized in the Table 5.2. The Δ notation in the table means the percentage change in communication efficiency.

Communication schemes	Δ no churn	Δ churn
Combination-multicast	0%	2%
Combination-unicast	0%	2%
Upward-unicast	0%	0%

Table 5.2: Percentage changes in communication efficiency in various communication schemes

Discussion and Analysis. The high values of ratio of successful communication in both high churn and no churn conditions along with the low standard deviation values show the efficiency and the stable of all evaluated communication schemes: combination-multicast, combination-unicast and upward-unicast.

On the other hand, the slightly changes or even no changes in ratio of successful communication in evaluated communication schemes (see Table 5.2) as the scale of the system increase 4 times, i.e. the number of parent MDMSs increase from 10 to 40 while the number of children MDMS increase from 500 to 2000, shows the scalability of the communication architecture.

5.5.2.5 Experiment results: Maintenance traffic and Scalability

In this work, we extend Chord DHT to implement MDMS overlays. Therefore, in this section, we evaluate the maintenance traffic of the proposed architecture by comparing the traffic generated by a parent peer and a children peer with the traffic generated by a Chord peer in the same condition. In the Figure 5.9, the two lines "parent vs. chord" and "child vs. chord" show the ratio of the traffic generated by a parent peer and a child peer respectively over the traffic generated a Chord peer in the same size network in churn condition.

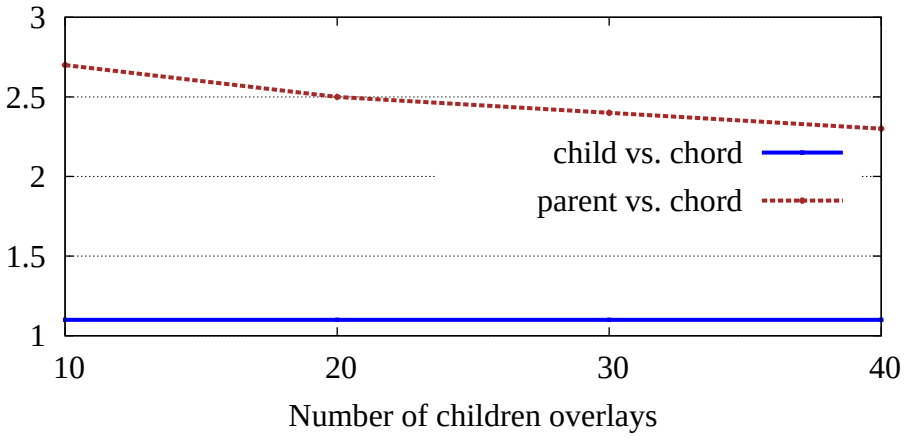

Figure 5.9: Average traffic generated by parents and children peers.

From the Figure 5.9, we can see that the line "child vs. chord" is horizontal at the value of 1.1. The line "parent vs. chord" is slightly decreases from 2.8 to 2.3 as the number of children overlay increase from 10 to 40. The standard deviations are approximately 0 in all cases.

Discussion and Analysis. The fact that the traffic generated by a child peer based on Chord equals 1.1 times the traffic generated by a Chord peer in the same size network, shows that additional traffic generated in a child peer is very small.

We note that the maintenance traffic of a peer, running DHT protocol, increases when the number of peers in the DHT overlay increases. In our experiment, when the number of children overlay increases from 10 to 40, i.e. the number of parent peer also increases, the line "parent vs. chord" is slightly decrease. This means that the maintenance traffic of parent overlay increases with slower rate than the maintenance traffic of Chord overlay as the size of the overlay increases. This proves the scalability of parent overlay in term of maintenance traffic.

Experiment results: Summary. The experiments shows that our proposed communication architecture achieves *high communication efficiency* and *low maintenance traffic* for maintaining children overlays. The architecture also features by the *scalability* in term of both the communication efficiency and the maintenance traffic.

5.6 Chapter summary

In this chapter, we have introduced a new AMI communication architecture for scalable data collection and management. The architecture is mixing of P2P and client-server model in which the MDMSs are geographically distributed and hierarchically organized in a P2P manner. The MDMSs at highest level play the roles of server managing data concentrators.

The analysis shows that the AMI based on our communication architecture is scalable for data collection and management while being resilient with failure and allow reducing the operation cost. Utility can also plan the geographic distribution of MDMSs to have expected latency of collecting data. Smart Grid applications can be deployed in OCs at different levels depending on their characteristics and requirements on latency.

The experiments shows the efficiency of communication schemes in the proposed architecture in term of ratio of successful communication in both high churn and no churn environments. The communication is performed with at least 97% of success under the high churn and with 100% of success in no churn condition. The experiments also show that the children peers generate low maintenance traffic, i.e. 1.1 times larger than a Chord peer in the same size network. The scalability of the communication architecture is shown in both aspects: communication efficiency and maintenance traffic.

This chapter relies on following work:

1. Giang Ngo Hoang, Luigi Liquori and Hung Nguyen Chan. A Scalable Communication Architecture for Advanced Metering Infrastructure, Technical Report.

Chapter 6

Concluding Remarks and Directions for Further Work

Contents

6.1 Conclusions

This thesis documented our work on two topics in which we exploit the inter-connecting of P2P overlay networks to allow the cooperation between P2P systems based on those networks and to establish the communication infrastructure for cyber-physical systems respectively.

Cooperation of P2P systems. With many P2P systems currently co-exist and compete with each other in different domains such as P2P file-sharing, instant messaging, P2P data storage, the cooperation to share the resources or services between them is expected. This motivated us to investigate the cooperation between heterogeneous P2P systems.

In this thesis, we introduced a *cooperation framework* for cooperation between heterogeneous P2P systems. The framework constitutes two parts. The first one is the OGP routing framework which allows the inter-overlay routing to enable the communication across P2P systems. The second part is *cooperation application*, which is built upon the routing infrastructure across P2P systems, bridging these P2P systems at application level.

The OGP routing framework allows to form an OGP overlay which conveys messages back and forth between heterogeneous P2P overlays including both structured and unstructured ones. The routing inside a P2P overlay follows the protocol native to that overlay while the routing on the OGP overlay follows the OGP protocol.

As different application domain has its own semantic, we could not introduce a framework for *cooperation application* for all application domains. Instead, we introduced principles underpinning the building of *cooperation application* which is applicable to all application domains. The separation of inter-overlay routing function and inter-application bridging function is the fundamental different between current work and others in which the transcoding of messages between different P2P systems is "built in" in the routing protocols. The new approach introduced by current work bring the great flexibility to the cooperation model thus makes the model more practical. Current model also allows the backward compatible in the sense that the peers, who are unaware of new

protocols, still work normally. This is also an important feature which makes the model more practical.

As a case study, we introduce a complete solution for co-operation of heterogeneous P2P file sharing networks. The *cooperation application*, IFP protocol, defines a set of intermediary messages. The transcoding of messages between format of P2P file sharing networks and intermediary format happen when the messages cross the border of the P2P file-sharing networks. Our solution ensures backward compatible and is applicable for all current P2P file-sharing networks.

We evaluated the inter-overlay routing framework and the cooperation solution for P2P file-sharing networks via experiment on large scale French's Grid 5000 platform.

Inter-connecting DHT routing infrastructure. In the second topic of this thesis, we investigate the ability of using the inter-connecting P2P overlays as routing infrastructure for cyber-physical systems via a case study in which inter-connecting DHTs serves as the fundamental part of the AMI communication infrastructure in smart grid.

In the vision of smart grid, AMI faces the challenge of scalable collecting and managing a huge amount of data from a large number of customers. In this topic, we address this challenge by proposing to use a communication architecture composed by the mixing of P2P and client-server models. In the architecture, the MDMSs of utility are geographically distributed and hierarchically connected by inter-connecting DHT overlays. The MDMSs at highest level collect the metering data from smart meters following the client-server model. Via analysis and experiment on large scale platform Grid5000, we show that the proposed architecture featured by following characteristics: scalability in collecting and management data, resilience with failure, scalability and efficiency in communication.

6.2 Further Work

There are many further works remain in the directions of this thesis.

Cooperation of real P2P file-sharing networks. In this thesis, we introduced a *cooperation solution* to cooperate P2P file-sharing networks and evaluated the solution with simple prototypes of three typical P2P file-sharing networks. However, cooperation of real P2P file-sharing networks requires more investigation. For example, the qualification in term of bandwidth and processing capacity, that a peer must have to be a FOGP peer, need to be specified. Another work is to propose schemes for dealing with the flooding request attack toward FOGP peers. An investigation about the efficient data retrieval when multiple data sources are available would be improving the performance of the system.

A framework for *cooperation application*. In this thesis, we proposed the principles underpinning the build of *cooperation application*. The next step could be a framework of *cooperation application* for bridging P2P applications in typical application domains such as P2P file-sharing, P2P instant messaging, P2P storage. One approach is to build this framework as the integration of multiple frameworks that each of which is for an application domain. Another approach is to build a common interface for multiple application domains and define the mapping between specific interface of each application domain and the common interface.

Cooperation of real instant messaging systems. There are many instant messaging networks with incompatible instant messaging protocols [IM]. Currently, to have these networks cooperate, one can combine the many disparate protocols inside the IM client application or inside the IM server application. With this approach, the changes in one protocol result in the changes in every multi-protocol IM client or multi-protocol IM server which implements the changed protocol. Our model provides another promising solution. Similar to cooperation of real P2P file-sharing networks, the cooperation of heterogeneous instant messaging systems requires a *cooperation application* which bridges these instant messaging systems at application level. Using our approach, an IM client or an IM server need to implement only one IM protocol and the cooperation application thus eliminates the need of changing when other protocols change.

Cloud-based applications. Cloud systems such as Amazon EC2, or NoSql databases, such as Amazon SimpleDB [simpleDB] or Cassandra [cassandra] or Google BigTable [Chang et al 2006] usually rely on a computer cluster; they offer complete control of the computing resources, such as CPUs or memory. The OGP framework can be used to form a routing infrastructure over the existing cloud systems while the cooperation applications on top of the OGP framework enable the exchanging data between these systems and resolves incompatibilities. While the performances will be surely affected by network artifacts, the scalability of this solution and the advantages of using overlay network federation make this approach attractive.

Cooperation of P2P clouds. Many studies such as [Babaoglu et al 2012, Kavalionak et al 2012] investigated the combination between cloud computing and P2P networks in which cloud is built upon the P2P network to take the advantages of P2P such as scalability, failure resilience, self-organization. These P2P clouds can achieve cooperation by relying on the OGP framework for establishing an routing infrastructure throughout the clouds and using a cooperation application for cooperating these clouds at application level.

Exploiting the inter-connecting overlays in cyber-physical systems. There are many cyber-physical systems in various domains. They often expand in large geographic areas and require scalable communication architectures for dealing with the intensive data generated by their sensors and actuators. Examples of these systems include autonomous automotive systems, medical monitoring, sensor-based communication-enabled autonomous, process control systems and distributed robotics. The inter-connecting overlays architecture with the features such as scalability, failure resilience, self-organization, allowing locality and heterogeneity can address mentioned challenge in cyber-physical systems.

Bibliography

[Camar 2009] G. Camarillo, Ed. *Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability.* Request for Comments: 5694, 2004. (Cited on page 5.)

[Andro et al 2004] S. Androutsellis-Theotokis and D. Spinellis. *A Survey of Content Distribution Technologies.* ACM Computing Surveys, 36(4), December 2004. (Cited on page 5.)

[Schol 2002] R. Schollmeier. *A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications.* Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002). (Cited on page 6.)

[Oram 2001] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* Online at http://oreilly.com/catalog/peertopeer/chapter/ch01.html. Accessed June 2013. (Cited on page 6.)

[Clark et al 2001] I.Clarke and O. Sandberg and B. Wiley and T. Hong. *Freenet: a distributed anonymous information storage and retrieval system.* International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability, 2001, Springer-Verlag New York, Inc. (Cited on page 9.)

[Miloj et al 2002] D. S. Milojicic and V. Kalogeraki and R. Lukose and K. Nagaraja and J. Pruyne and B. Richard and S. Rollins and Z. Xu *Peer-to-Peer Computing.* HP Laboratories Palo Alto, HPL-2002-57, 2002. (Cited on page 11.)

[Usenet] Wikipedia. *Usene.* Online at http://en.wikipedia.org/wiki/Usenet. Accessed June 2013. (Cited on page 6.)

[Napst] Wikipedia. *Napster.* Online at https://en.wikipedia.org/wiki/Napster. Accessed June 2013. (Cited on page 7.)

[Gnute] *Gnutella.* Online at http://rfc-gnutella.sourceforge.net/developer/index.html. Accessed June 2013. (Cited on pages 7 and 9.)

[Kazaa] *Kazaa.* Online at http://en.wikipedia.org/wiki/Kazaa. Accessed June 2013. (Cited on pages 7 and 9.)

[eDonk] *eDonk.* Online at http://en.wikipedia.org/wiki/EDonkey_network. Accessed June 2013. (Cited on pages 7 and 9.)

[Kad] *Kad.* Online at http://en.wikipedia.org/wiki/Kad_network. Accessed June 2013. (Cited on page 7.)

[BitTo] *BitTo.* Online at http://en.wikipedia.org/wiki/BitTorrent. Accessed June 2013. (Cited on pages 7 and 9.)

[Babao 2012] O. Babaoglu. *Introduction to Peer-to-Peer Systems.* Complex Systems. UniversitÃ¡ di Bologna. Retrieved 12 June 2013. (Cited on page 7.)

[DNS] Wikipedia. Online at http://en.wikipedia.org/wiki/Domain_Name_System. Accessed June 2013. (Cited on page 7.)

[Rfc1034] P. Mockapetris. Online at http://tools.ietf.org/html/rfc1034. 1987, Accessed June 2013. (Cited on page 7.)

[Rfc1035] P. Mockapetris. Online at http://tools.ietf.org/html/rfc1035. 1987, Accessed June 2013. (Cited on page 7.)

[Livestation] Wikipedia. Online at http://en.wikipedia.org/wiki/Livestation Accessed June 2013. (Cited on page 14.)

[TVUplayer] Wikipedia. Online at http://en.wikipedia.org/wiki/TVUnetworks Accessed June 2013. (Cited on page 14.)

[PPLive] Wikipedia. Online at http://en.wikipedia.org/wiki/PPLive Accessed June 2013. (Cited on page 14.)

[Zattoo] Wikipedia. Online at http://en.wikipedia.org/wiki/Zattoo Accessed June 2013. (Cited on page 14.)

[Small-world] Wikipedia. Online at http://en.wikipedia.org/wiki/Small-world_network Accessed June 2013. (Cited on page 13.)

[JXTA] Wikipedia. Online at http://en.wikipedia.org/wiki/JXTA Accessed June 2013. (Cited on page 18.)

[Tit-for-tat] Wikipedia. Online at http://en.wikipedia.org/wiki/Tit_for_tat Accessed June 2013. (Cited on page 17.)

[BBC-iPlayer] BBC. Online at http://www.bbc.co.uk/iplayer/ Accessed June 2013. (Cited on page 14.)

[Bitcoin] Wikipedia. Online at http://en.wikipedia.org/wiki/Bitcoin Accessed June 2013. (Cited on page 18.)

[Azureus] Vuze. Online at http://www.vuze.com/ Accessed June 2013. (Cited on page 17.)

[BitTorrent] BitTorrent. Online at http://www.bittorrent.com/ Accessed June 2013. (Cited on page 17.)

[μTorrent] μTorrent website. Online at http://www.utorrent.com/ Accessed June 2013. (Cited on page 17.)

[Bitcomet] Bitcomet website. Online at http://www.bitcomet.com/ Accessed June 2013. (Cited on page 17.)

[Tribler] Tribler website. Online at http://www.tribler.org/trac Accessed June 2013. (Cited on page 17.)

[Buford et al 2010] J. Buford and H. Yu. *Peer-to-Peer Networking and Applications: Synopsis and Research Directions.* Handbook of Peer-to-Peer Networking, 2010. (Cited on page 14.)

[Gupta et al 2002] I. Gupta and K. Birman and P. Linga and A. Demers and R. v. Renesse. *Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead.* Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), 2003. (Cited on page 11.)

[Monnerat et al 2006] L. Monnerat and C. Amorim. *D1HT: A Distributed One Hop Hash Table.* Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2006. (Cited on page 11.)

[Kaashoek et al 2003] M. F. Kaashoek and D. R. Karger. *Koorde: A simple degree-optimal distributed hash table.* Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), 2003. (Cited on page 11.)

[Li et al 2005] J. Li and J. Stribling and R. Morris and M. F. Kaashoek. *Bandwidth-efficient management of DHT routing tables.* Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2003. (Cited on page 11.)

[Brown et al 2009] A. Brown and M. Kolberg and J. Buford. *Chameleon: An Adaptable 2-Tier Variable Hop Overlay.* Proceedings of the Sixth Consumer Communications and Networking Conference (CCNC), 2009. (Cited on page 11.)

[Alima et al 2003] L.O. Alima and S. El-Ansary and P. Brand and S. Haridi. *DKS(N, k, f): a family of low communication, scalable and fault-tolerant infrastructures for P2P applications.* Proceedings of thrid IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), 2003 . (Cited on page 11.)

[Zhao et al 2004] B. Y. Zhao and L. Huang and J. Stribling and S. C. Rhea and A. D. Joseph and J. D. Kubiatowicz. *Tapestry: A Resilient Global-scale Overlay for Service Deployment.* IEEE Journal on Selected Areas in Communications (JSAC), 2004. (Cited on page 11.)

[Schlosser et al 2003] M. Schlosser and M. Sintek and S. Decker and W. Nejdl. *HyperCuP: hypercubes, ontologies, and efficient search on peer-to-peer networks.* Proceedings of the 1st international conference on Agents and peer-to-peer computing (AP2PC), 2003. (Cited on page 12.)

[Malkhi et al 2002] D. Malkhi and M. Naor and D. Ratajczak. *Viceroy: a scalable and dynamic emulation of the butterfly.* Proceedings of the twenty-first annual symposium on Principles of distributed computing (PODC), 2002. (Cited on page 12.)

[Aberer et al 2003] K. Aberer and CP. udré-Mauroux and A. Datta and Z. Despotovic and M. Hauswirth and M. Punceva and R. Schmidt. *P-Grid: a self-organizing structured P2P system.* ACM SIGMOD Record, 2003. (Cited on page 12.)

[Ripea et al 2002] M. Ripeanu, I. Foster, and A. Iamnitchi. *Mapping The Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems And Implications For System Design.* IEEE Internet Computing Journal, 6(1), 2002. (Cited on page 9.)

[Marka et al 2002] E. P.Markatos. *Tracing a Large-Scale Peer to Peer System: AnHour in the Life of Gnutella.* In The Second International Symposium on Cluster Computing and the Grid, 2002. (Cited on page 9.)

[Cherv et al 2008] A. Chervenak and S. Bharathi. *Peer-to-peer Approaches to Grid Resource Discovery.* Making Grids Work: Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System Architecture Grid Systems, Tools and Environments 12-13 June 2007, Heraklion, Crete, Greece. Springer. p. 67. ISBN 9780387784489 (Cited on page 9.)

[Jin et al 2010] X. Jin and S.-H. Chan. *Unstructured Peer-to-Peer Network Architectures.* In Shen et al. Handbook of Peer-to-Peer Networking. Springer. p. 119. ISBN 978-0-387-09750-3, 2010. (Cited on page 9.)

[Lv et al 2002] Q. Lv and S. Ratsnasamy and S.Shenker *Can Heterogenity Make Gnutella Stable?.* IPTPS 2002. (Cited on pages 9 and 12.)

[Hui et al 2006] K. Hui and J. Lui and D. Yau. *Small-world overlay P2P networks: construction, management and handling of dynamic flash crowds.* Computer Networks, 2006. (Cited on page 13.)

[Ganes et al 2003] P. Ganesan and Q. Sun and H. Garcia-Molina. *Yappers: A peer-to-peer lookup service over arbitrary topology.* Infocom, 2003. (Cited on page 13.)

[Garce et al 2003] L. Garces-Erce and K.W. Ross and E. Biersack and P. Felber and G. Urvoy-Keller. *TOPLUS: Topology Centric Lookup Service.* Fifth International Workshop on Networked Group Communications (NGC) 2003. (Cited on page 14.)

[Xu et al 2003] Z. Xu and R. Min and Y. Hu. *HIERAS: a DHT based hierarchical P2P routing algorithm.* International Conference on Parallel Processing (ICPP), 2003. (Cited on page 14.)

[Maymo et al 2002] P. Maymounkov and D. Mazières. *Kademlia: A Peer-to-peer Information System Based on the XOR Metric.* Proc. of IPTPS, 2002. (Cited on pages 11, 12 and 28.)

[Rowstron et al 2001] A. I. T. Rowstron and P. Druschel. *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.* Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, 2001. (Cited on page 11.)

[Ratna et al 2001] S. Ratnasamy and P. Francis and M. Handley and R. Karp and S. Shenker. *A scalable content-addressable network.* Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '01. (Cited on page 11.)

[Stoic et al 2001] I. Stoica and R. Morris and D. Karger and M.F. Kaashoek and H. Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications.* SIGCOMM, 2001 (Cited on pages 11, 82 and 85.)

[Artig at al 2005] M. Artigas and P. Lopez and J. Ahullo and A. Skarmeta. *Cyclone: A Novel Design Schema for Hierarchical DHTs.* Proc. of the Fifth IEEE International Conference on Peer-to-Peer Computing, 2005. (Cited on page 14.)

[Ganes at al 2004] P. Ganesan and K. Gummadi and H. Garcia-Molina. *Canon in G Major: Designing DHTs with Hierarchical Structure.* IEEE International Conference on Distributed Computing Systems (ICDCS), 2004. (Cited on page 14.)

[Castro at al 2005] M. Castro and M. Costa and A. Rowstron. *Debunking some myths about structured and unstructured overlays.* Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI), 2005. (Cited on page 13.)

[Seti] *SETI@HOME*. Online at http://www.seti.org/. Accessed June 2013. (Cited on page 17.)

[AppUsage] *Application usage and threat.* Online at http://researchcenter.paloaltonetworks.com/app-usage-risk-report-visualization/#. Accessed June 2013. (Cited on page 16.)

[RaTiOn 2001] Ra Ti On. *Project JXTA: An Open, Innovative Collaboration.* (Cited on page 18.)

[LatestNews] *JXTA*. Online at https://kenai.com/projects/jxse/pages/LatestNews. Accessed June 2013. (Not cited.)

[Skype] *Skype*. Online at http://www.skype.com/en/. Accessed June 2013. (Cited on page 17.)

[Baset at al 2006] S.A. Baset and H.G. Schulzrinne. Online at AnAnalysisoftheSkypePeer-to-PeerInternetTelephonyProtocol. INFOCOM 2006. (Cited on page 17.)

[Y!M] *Yahoo! Messenger*. Online at http://messenger.yahoo.com/. Accessed June 2013. (Cited on page 17.)

[Pidgin] *Pidgin*. Online at http://www.pidgin.im/. Accessed June 2013. (Cited on page 17.)

[Konis at al 2006] J. Konishi and N. Wakamiya and M. Murata. *Proposal and evaluation of a cooperative mechanism for pure p2p file sharing networks.* Proc. of BioADIT, 2006 (Cited on pages 19, 20, 47, 48 and 71.)

[Cheng 2007] L. Cheng. *Bridging Distributed Hash Tables in Wireless Ad-Hoc Networks.* GLOBECOM, 2007 (Cited on pages 19, 20, 47 and 48.)

[Liquo at al 2009] L. Liquori and C. Tedeschi and F. Bongiovanni. *Babelchord: a social tower of DHT-based overlay networks.* ISCC, 2009 (Cited on pages 19, 20, 47 and 48.)

[Liquo at al 2010] L. Liquori and C. Tedeschi and L. Vanni and F. Bongiovanni and V. Ciancaglini and B. Marinkovic. *Synapse: A Scalable Protocol for Interconnecting Heterogeneous Overlay Networks.* Networking, 2010 (Cited on pages 19, 20, 47 and 48.)

[Cianc at al 2012] V. Ciancaglini and L. Liquori and G. Ngo and P. Maksimovic. *An Extension and Cooperation Mechanism for Heterogeneous Overlay Networks.* Networking Workshops, 2012 (Cited on pages 19, 20, 47 and 48.)

[Cianc et al 2011] V. Ciancaglini and L. Liquori and G. Ngo. *Towards a Common Architecture to Interconnect Heterogeneous Overlay Networks.* The 17th International Conference on Parallel and Distributed Systems, ICPADS 2011, pp. 817-822, Tainan, Taiwan, December 7-9, 2011. (Cited on pages 19, 20, 47 and 48.)

[Kwon at al 2005] M. Kwon and S. Fahmy. *Synergy: an overlay internetworking architecture.* Proc. of ICCCN, 2005 (Cited on page 21.)

[Jiang at al 2005] W. Jiang, D. Chiu and J. Lui. *On the Interaction of Multiple Overlay Routing.* Performance Evaluation, 62(1-4), 2005 (Cited on page 21.)

[Liao et al 2006] X. Liao, H. Jin, Y. Liu, L. Ni and D. Deng. *AnySee: Peer-to-Peer Live Streaming.* INFOCOM 2006. (Cited on page 22.)

[Datta at al 2006] A. Datta and K. Aberer. *The Challenges of Merging Two Similar Structured Overlays: A Tale of Two Networks.* Proc. of EuroNGI 2006 (Cited on page 21.)

[Shafa at al 2009] T. M. Shafaat and A. Ghodsi and S. Haridi. *Dealing with network partitions in structured overlay networks.* Peer-to-Peer Networking and Applications, 2009 (Cited on page 21.)

[P2pfs] *Comparison of file sharing applications.* Online at `http://en.wikipedia.org/wiki/Comparison_of_file_sharing_applications`. Accessed June 2013. (Not cited.)

[IM] *Comparison of instant messaging protocols.* Online at `http://en.wikipedia.org/wiki/Comparison_of_instant_messaging_protocols`. Accessed June 2013. (Cited on page 93.)

[G5k] *Grid5000 platform.* Online at `https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home`. Accessed June 2013. (Cited on pages 42, 65, 74 and 86.)

[simpleDB] *Amazon simpleDB.* Online at `http://aws.amazon.com/simpledb/`. Accessed June 2013. (Cited on page 93.)

[cassandra] *The Apache Cassandra Project.* Online at `http://cassandra.apache.org/`. Accessed June 2013. (Cited on page 93.)

[BGP] Y. Rekhter and T. Li *A Border Gateway Protocol 4 (BGP-4).* Online at `http://tools.ietf.org/html/rfc4271`. Accessed June 2013. (Not cited.)

[Castro et al 2006] M. Castro and P. Druschel and A-M . Kermarrec and A. I.T. Rowstron. *Scribe: a large-scale and decentralized application-level multicast infrastructure.* IEEE J.Sel. A. Commun. 2006 (Cited on page 84.)

[El-Ans et al 2003] S. El-Ansary and L. Alima and P. Brand and S. Haridi. *Efficient Broadcast in Structured P2P Networks.* IPTPS 2003 (Cited on pages 84 and 85.)

[Ngo et al 2013] G. Ngo and L. Liquori and V. Ciancaglini and P. Maksimovic and H. Chan. *A Backward-Compatible Protocol for Inter-routing over Heterogeneous Overlay Networks.* Proc. of SAC 2013 (Not cited.)

[Chang et al 2006] F. Chang and J. Dean and S. Ghemawat and Wilson C. Hsieh and D. Wallach and M. Burrows and T. Chandra and A. Fikes and R. Gruber. *Bigtable: a distributed storage system for structured data.* Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI) 2006 (Cited on page 93.)

[ami 2008] US Dept. of Energy, Office of Electricity Delivery and Energy Reliability. *Advanced Metering Infrastructure.* 2008 (Cited on pages 74, 75 and 76.)

[Rusit et al 2009] S. Rusitschka and C. Gerdes and K. Eger. *A low-cost alternative to smart metering infrastructure based on peer-to-peer technologies.* 6th International Conference on the European Energy Market, 2009. EEM 2009. 2009. (Not cited.)

[Jiazh et al 2012] Z. Jiazhen and R.Q. Hu and Y. Qian. *Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid.* IEEE Transactions on Parallel and Distributed Systems. 2012. (Cited on page 76.)

[Gerdes et al 2009] C. Gerdes and U. Bartlang, and J. Muller. *Vertical Information Integration for Cross Enterprise Business Processes in the Energy Domain.* Proceedings of Agent-Based Technologies and Applications for Enterprise Interoperability. 2009. (Cited on page 76.)

[Meili et al 2013] S. Meiling and T. Steinbach and T. C. Schmidt and M. Wählisch. *A Scalable Communication Infrastructure for Smart Grid Applications using Multicast over Public Networks.* Proc. of ACM SAC. 2013. (Cited on page 76.)

[Arena et al 2010] M. Arenas-Martinez and S. Herrero-Lopez and A. Sanchez and J.R. Williams and P. Roth and P. Hofmann and A. Zeier. *A Comparative Study of Data Storage and Processing Architectures for the Smart Grid.* Proc. of SmartGridComm. 2010. (Cited on page 76.)

[Jagadish et al 2005] H. V. Jagadish and B. C. Ooi and K.L. Tan and Q. H. Vu and R. Zhang. *BATON: a balanced tree structure for peer-to-peer networks.* Proceedings of the 31st international conference on Very large data bases. 2005. (Cited on page 76.)

[Leitao et al 2007] J. Leitao and J. Pereira and L. Rodrigues. *Epidemic Broadcast Trees.* Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems. 2007. (Cited on page 76.)

[Li et al 2006] M. Li and W.c. Lee and A. Sivasubramaniam. *DPTree: A Balanced Tree Based Indexing Framework for Peer-to-Peer Systems.* Proceedings of the IEEE International Conference on Network Protocols. 2006. (Cited on page 76.)

[Stand 2010] *SmartGrid/AEIC AMI interoperability standard guidelines for ANSI C12.19 / IEEE 1377 / MC12.19 end device communications and supporting enterprise devices, networks and related accessories.* The Association of Edison Illuminating Companies, Meter and Service Technical Committee report version 2. 2010. (Cited on page 74.)

[Queen 2011] E. Queen. *A discussion of smart meters and RF Exposure Issues.* Edison Electric Institute (EEI). 2011. (Not cited.)

[Bernaudo et al 2010] D. Bernaudo et al. *SmartGrid/AEIC AMI interoperability standard guidelines for ANSI C12.19 / IEEE 1377 / MC12.19 end device communications and supporting enterprise devices, networks and related accessories.* Online at `http://www.aeic.org/meterservice/`

AEICSmartGridStandardv2-11-19-10.pdf. The Association of Edison Illuminating Companies, Meter and Service Technical Committee report version 2. 2010. (Cited on page 74.)

[Tram 2011] H. Tram *Technical and operation considerations in using Smart Metering for outage management.* Transmission and Distribution Conference and Exposition. IEEE/PES. 2008. (Not cited.)

[Cascio] J. Cascio *Resilience.* Online at http://www.fastcompany.com/blog/jamais-cascio/open-future/resilience. Accessed September 2013. (Cited on page 81.)

[Scala] Wikipedia. *Scalability.* Online at http://en.wikipedia.org/wiki/Scalability. Accessed June 2013. (Cited on page 80.)

[CPS] Wikipedia. *Cyber Physical System.* Online at http://en.wikipedia.org/wiki/Cyber-physical_system. Accessed June 2013. (Cited on pages 2 and 73.)

[AMI] Wikipedia. *Advanced Metering Infrastructure.* Online at http://en.wikipedia.org/wiki/Smart_meter. Accessed June 2013. (Cited on page 2.)

[CDN] Wikipedia. *Content Delivery Network.* Online at http://en.wikipedia.org/wiki/Content_delivery_network. Accessed June 2013. (Cited on page 76.)

[PubSub] *Publish Subscribe pattern.* Online at http://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern. Accessed Oct 2013. (Cited on page 79.)

[CPS] *Cyber Physical Systems.* Online at http://cyberphysicalsystems.org/. Accessed Oct 2013. (Cited on pages 2 and 73.)

[Cisco 2009] *Cisco definition of Smart Grid..* Online at http://www.cisco.com/web/strategy/docs/energy/CISCO_IP_INTEROP_STDS_PPR_TO_NIST_WP.pdf. Accessed July 2013. (Cited on page 73.)

[Hart 2008] D. G.Hart. *Using AMI to Realize the Smart Grid.* IEEE, 2008. (Not cited.)

[Rusitschka et al 2010] S. Rusitschka and K. Eger and C. Gerdes. *Smart Grid Data Cloud: A Model for Utilizing Cloud Computing in the Smart Grid Domain.* Proceedings of IEEE First Int'l Conf. Smart Grid Comm, 2010. (Cited on page 76.)

[Khan et al 2013] R. H. Khan and J. Y. Khan. *A comprehensive review of the application characteristics and traffic requirements of a smart grid communications network.* Computer Networks, 2013. (Cited on page 74.)

[Babaoglu et al 2012] O. Babaoglu and M. Marzolla and M. Tamburini. *Design and implementation of a P2P Cloud system.* Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC), 2012. (Cited on page 93.)

[Kavalionak et al 2012] H. Kavalionak and A. Montresor. *P2P and cloud: a marriage of convenience for replica management.* Proceedings of the 6th IFIP TC 6 international conference on Self-Organizing Systems (IWSOS), 2012. (Cited on page 93.)