Proceedings of the IASTED International Conference
Advances in Computer Science (ACS 2013)
April 10 - 12, 2013 Phuket, Thailand

A CASE STUDY OF HOME SERVICE SHARING USING RELOAD

Warodom Werapun and Apichat Heednacram
Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University, Thailand
email: {warodom, apichat}@coe.psu.ac.th

ABSTRACT
In present days, home devices with network capabilities are widely used and augmented for delivering several Home Services (HS) to more end users. This technology increases user experiences in sharing services by accessing services locally, remotely or visiting other user services. The interesting research problem is how to organize the service sharing reliably and effectively while the service cost and speed are still acceptable. We focus on Structured Peer to Peer (P2P) network architecture in comparison with other two types of P2P architecture, namely Centralized P2P and Pure P2P. The comparison is presented in terms of signaling overhead and service specifications. We have previously studied the Centralized P2P and Pure P2P network architecture for HS but never Structured P2P. The Structured P2P uses Distributed Hash Table (DHT) to quickly lookup and route shared services in an overlay network. In this paper, we propose to use RELOAD (REsource LOcation And Discovery Base Protocol) which provides an abstract storage in the P2P overlay, as network architecture for managing HS sharing in the decentralized way. Our evaluation results reveal that HS sharing using RELOAD offers several advantages and the architecture is suitable for our case study, the weather report service sharing.

KEY WORDS
Home Service, RELOAD, P2P, Network architecture

1. Introduction

With the growth of the Internet, many homes usually have network connections as one of the mandatory things in life. Home equipments such as televisions, refrigerators, doors, light controllers and air conditions come with embedded network capabilities for creating a home automation and connecting to other ubiquitous devices (e.g., digital camera, mobile phones).

Next generation of home equipments and ubiquitous devices are expected to provide more and better of these services. For example, users can display a list of food in a refrigerator from its own service, control the lights, or share their photos in a digital camera to other display devices (e.g., a television). When these services are executed at home, especially home equipments, they are defined as Home Services (HS). Moreover, HS can be shared between several homes to increase user benefits which depend on the service owner policies. As a result, HS will be undoubtedly used more in up-and-coming modern communities. Therefore, the challenging problem is to manage a number of sharing HS as service communities that offer high reliability, high efficiency, high service speed but low service cost.

Our previous investigation has been done in Centralized P2P SIP solution [1] and distributed Pure P2P solution where authentication mechanisms are proposed in our works [2, 3]. Another distributed P2P solution called DHT P2P is also important and therefore in this paper we consider DHT P2P studied by IETF which is one of the latest P2P generations called RELOAD [4] in comparison with the previous network architectures [2, 3]. However, RELOAD technology is still in working progress.

To the best of our knowledge, RELOAD has never been used to deliver shared HS before. We, therefore, propose to use RELOAD as an abstract storage to index shared HS. To test and evaluate our new concept, we provide a case study of "weather report service sharing" as a home service. We evaluate the benefits in terms of signaling overhead and the comparison of service specifications. Our contribution includes the design of RELOAD architecture for HS, the implementation of whether report service sharing using Java language, and the useful discussion of good points and bad points among three architectures (i.e., RELOAD, Pure and Centralized).

This paper is organized as follows. In Section II we present an overview of RELOAD and a topology for HS sharing. In Section III, we design and implement HS sharing using RELOAD based on the case study. In Section IV we present our evaluation results and analysis. Finally, in Section V we summarize and conclude with open problems.

2. Technical Background

RELOAD uses Session Initiation Protocol (SIP) as a based signaling protocol. Therefore, we will explain SIP briefly before giving the technical background of RELOAD.

2.1 Session initial protocol

SIP [1] is a text based signaling protocol to set up multimedia sessions between endpoints. These sessions may be texts, games, voices, videos or a combination of these sessions. SIP is similar to HTTP protocol. SIP locates users using Uniform Resource Identifier (URI)

DOI: 10.2316/P.2013.801-062

(e.g., sip:user@coe.psu.ac.th). SIP usually runs on port 5060 but users can select its transport protocol (e.g., UDP, TCP, SCTP, and so on). For describing the session, SIP often uses Session Description Protocol (SDP). If SIP is a control plane protocol, then delivery of the media, or a data plane, is commonly based on RTP (Real-time Transport Protocol).

2.2 RELOAD

RELOAD is a P2P signaling protocol which provides peers with an abstract storage in the P2P overlay. This storage is formed by collecting shared resources or services from connected nodes. RELOAD technology was introduced in 2008 [4]. Its utilisation is still under study and hence there is no RELOAD open source that provides enough documents and features for home service delivery. However, RELOAD appears in several research domains; for example, Yu et al. [5] used RELOAD for a secure P2P SIP communication for the public Internet with certificate based, and routing in NAT environment. Maenpaa and Bolonio [6] studied the increase in service availability for resource constrained nodes in P2P by proposing a novel overlay construction algorithm. RELOAD performance on mobile phone was also analyzed and application layer multicast extensions to RELOAD were proposed [7, 8].

RELOAD uses P2P Session Initial Protocol (P2PSIP) [9] network and it requires Chord [10] as a mandatory overlay routing protocol between peers. There are 3 main steps for Chord routing and network maintenance.

1. **Join**: To attach a node to an overlay network. Join step uses other processes that are Find Successor, Find Predecessor, Stabilized and Notify Successor respectively.

2. **Stabilized**: To update its Successor and the Predecessor of its Successor. This step is called periodically.

3. **Fix Finger Table**: To update Chord nodes inside Finger table in each Finger object. This step is also called periodically.

Once we have an abstract storage that is created by shared nodes, then nodes require 5 steps in order to access and share HS.

1. **Registering:** The sharer and downloader enroll themselves to the overlay network.

2. **Publishing:** The sharer maintains service indexes (address of shared services) to downloader.

3. **Searching:** The downloader looks up shared indexes.

4. **Retrieving:** The downloader gets resources from sharers directly.

To make it more practical before implementing this network architecture, we need to know how nodes are linked together in the context of shared HS since different topologies offer different service lookup costs. We discuss this point next.

2.3 P2P topology for home service sharing

Home network can be divided into two levels. The first level called intra-home network level is the communication between a Home Gateway (HGw) with all home devices. Topology in intra-home usually is the centralized architecture since there are few nodes directly connected with HGw. The HGw handles all connected home devices in order to provide HS and manages several HS as shown in Figure 1.

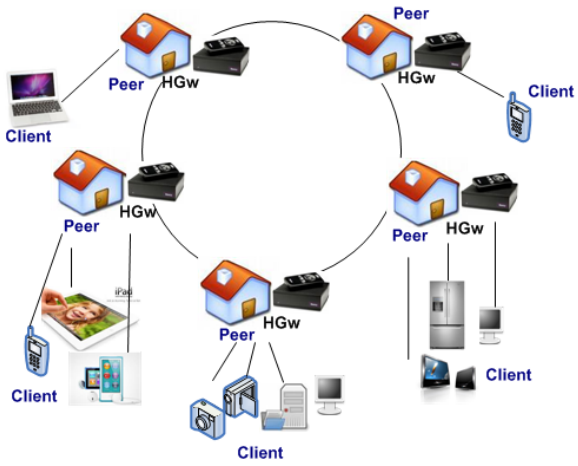

Figure 1. HS sharing on HGw overlay network.

The second level called inter-home network level is the communication among HGw. This level is more complicated than the intra-home level since there could be a number of HGw participated in the network. In this level, centralized or distributed solution is issued for service sharing to create HS communities.

In RELOAD, a node called a peer, handles routing messages to maintain an overlay. On the other hand, a client is able to attach a peer in order to access sharing services. A client will not manage routing messages while a peer does. This is a main difference between a peer and a client. We propose to define HGw as a peer element, and home devices as a client element because of following reasons:

- Home devices do not have appropriate network connectivity, e.g., connect behind the NAT typically.
- Home devices may not have sufficient resources, especially small ubiquitous devices, such as computing power, storage space, or battery power.
- The overlay algorithm may dictate specific requirements for peer selection. These may include participating in the overlay to determine trustworthiness; controlling a number of peers in the overlay to reduce overly-long routing paths; or

ensuring minimum application uptime before a node can join as a peer [4].

3. Home Services (HS) Application

In this section we introduce an application for service sharing. We will look at a case study of weather report service. This service will be deployed as "a home service sharing". We choose this HS since it is a simple service and can be used to illustrate HS sharing functionalities. All authenticated users can access sharing service, download the service, and propose the service to others. We begin with the service description, explore four main steps to access and share the service. We also explain an additional step of how to deploy the service. Eventually, our design and implementation architecture are described.

3.1 Description of weather report service sharing

In Figure 2, we define HGw as a peer and weather sensor controller as a client. In this scenario, Bob would like to share weather report service. Alice searches from a repository (a peer overlay), retrieves and installs this service to her device. Then, John can access a weather report service from Alice or Bob. Service sharing is a new idea, thus there are not many practical implementations because deploying service sharing is quite complicated. We explain steps of accessing and deploying service sharing in the next subsection.

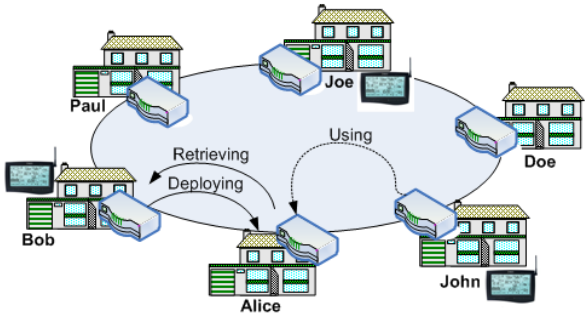

Figure 2. Weather report service sharing.

3.2 Steps of accessing and deploying weather report service sharing

There are four main steps of accessing weather report service. We add the deployment step as the fifth step to show how the service is completely activated on different homes.

3.2.1 Initializing

Sensors periodically capture weather information. Bob then deploys his service that periodically retrieves weather information from his sensors. Bob has to define service specifications for users who want to download the service. Bob's HGw has to join an overlay by pre-computing its identity to find its location or asks its identity from a trusted server for security reasons. Then, a client can be attached to HGw similar to Figure 1.

```
<index>
   <nodeId>621</nodeId>
   <nodeIP>147.127.240.90</nodeIP>
   <nodePort>5061</nodePort>
   <ResId>13974</ResId>
   <ResName>Weather Report Service</ResName>
   <ResSignature>iEYARA……QgkQJ9S</ResSignature>
</index>
```

Figure 3. An index example of service sharing.

3.2.2 Publishing

Bob publishes the service index (Figure 3) to an overlay by searching for a node that keeps a service index according to Chord algorithm. Bob hashes a keyword to an identity that corresponds to the published service.

3.2.3 Searching

Alice searches Bob's service. Similar to Publishing step, she hashes a keyword to an identity that has the same number of an identity published by Bob; look ups and downloads an index from the destination node. Then, she can download the service from Bob after knowing his location from the index.

3.2.4 Retrieving

In this step, SIP protocol is used after Alice knew the destination of preferred shared HS. Alice retrieves his service by sending *SIP INVITE* and waits for response *(200 OK)* from Bob. After she gets a response from Bob, she sends *ACK* and service retrieving request. Bob sends a service description to her in order to establish a session. Then, Alice can download shared HS. Figure 4 shows an example of the service description.

```
<service>
 <information>
    <Id>13974</Id>
    <name>Weather Report Service</name>
    <vendor>Warodom WERAPUN</vendor>
  </information>
  <resources>
    <!-- Application Resources -->
    <j2se version="1.6+"
href="http://java.sun.com/products/autodl/j2se"/
>
    <jar href="weather.jar" main="true" />
  </resources>
  <application-desc
    name="Weather Report Service"
    main-class="org.weather.Main"
    width="800"
    height="600">
  </application-desc>
  <update check="background"/>
</service>
```

Figure 4. A service description example.

3.2.5 Deploying

Assume that a service is deployed on Alice's home; Figure 5 shows a service deployment flowchart. From the flowchart, she must allow software installation from Bob. If she does not have a runtime, the system will download and install the runtime (e.g., JRE) for her. Then, the system will download object or executable files, validate integrity and install them on her device. After that her weather report service will be activated. In addition, she can propose this service to other users. For example, John can search this service and can use the service from a sharer (like Bob) or Alice. Since weather report service comes from Bob's home, Alice retrieve indirectly weather information from his sensors, then the service at Alice's home has to update weather information from Bob's service periodically. We note that in this deployment step, when a user downloads and installs a sharing service, a sharing service may be completely installed on that user device, or it is just simply an entry point linking to a service owner or some web servers or web service interfaces etc. These options will depend on the service policies since there are many types of services sharing services available.

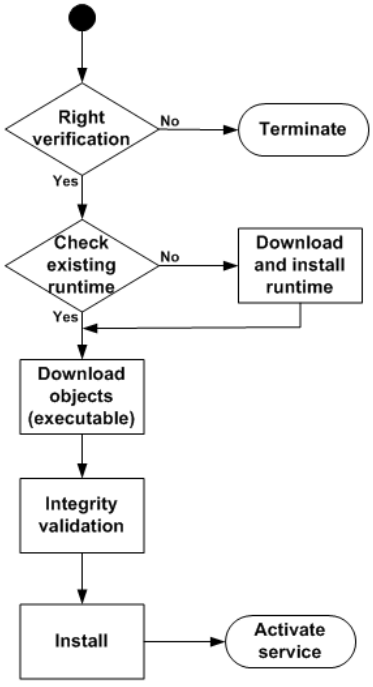

Figure 5. Service deployment flowchart.

3.3 Design

We design RELOAD system to show a proof of concept. RELOAD is fundamentally an overlay network. The main class diagram is as shown in Figure 6. We use Key object as a HashMap index and store Resource object that contains Chord nodes and arbitrary strings of service addresses (e.g., service names or URLs) into the HashMap. Every Chord node has Finger Table that keeps finger object elements for Chord routing.

3.4 Implementation architecture

Java language will be used for the implementation as it offers various useful libraries. It includes TLS/SSL, SIP and several storage collection libraries. RELOAD is a message oriented request/response protocol. In the routing layer, peers (HGw) communicate to each other by exchanging defined XML messages which are mapped with Chord Message class such as Plain Old Java Object (POJO). The POJO is a Java class that contains only class members and correspondent methods. Every Chord node has Finger Table that keeps finger object elements for chord routing. Predecessor node is stored in all finger objects contained in a finger table. This setting reduces not only searching time but also publishing time. In our implementation, TLS/SSL is used to protected overlay connection as defined in the standard.

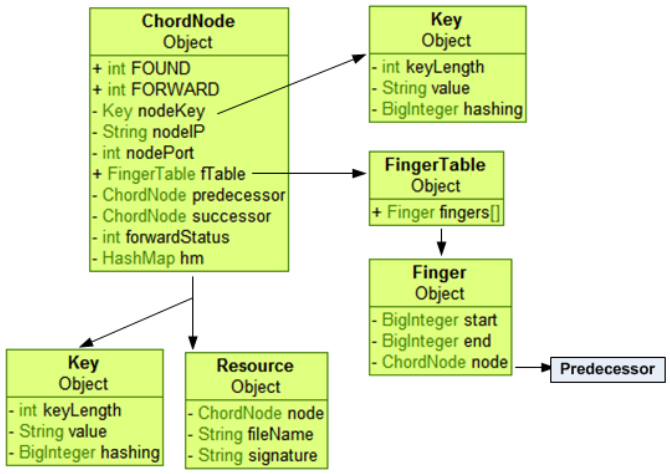

Figure 6. The main class diagram.

4. Evaluation Result

In this section, we compare signaling overhead that is used in three architectures; RELOAD, Pure P2P and Centralized P2P. Before we discuss and evaluate our simulation results, we will describe our experiment and our assumptions. The simulation is performed using 10,000 nodes acting as connected HGw and each HGw has10 nodes. For Centralized P2P network architecture, we assumed that the architecture has a centralized server communicating among nodes. In Pure P2P overlay, since there are many possible topologies, we will model a distribution of node degrees according to a power law [11]. An overlay of HGw has compliant topologies with the distribution of Jovanovic [12]. When a node *i* joins the network, the probability that it connects to a node *j* already belong to the network is given by *P(i,j)* as indicated in Equation (1), where *d* is the degree of the target node, *V* represents the entire set of nodes in the

network and $\sum_{k \in V} d_k$ is the sum of degrees of all nodes that have previously joined the network.

$$P(i,j) = \frac{d_j}{\sum_{k \in V} d_k} \quad (1)$$

The node degree is computed by using Equation (1). Equation (1) does not depend on *i* (a joining node) because the probability of its neighbour node depends on the degree of the previously-joined node (referring to the power law theory). In the implementation point of view, we use BRITE [13] as a tool to generate our topologies based on Equation (1)'s principle.

Flooding algorithm is used with TTL=4 as a default value of GNUtella for searching in Pure P2P overlay. In RELOAD, an HGw overlay is established by generating uniform random node identity of 10,000 nodes and form Chord ring as a mandatory DHT routing protocol.

Note that Internet is growing so fast and reaches more homes than ever before. Connected Home Networks (HN) in level 1,000 to 10,000 nodes, therefore, is an interesting size to investigate. If the performance of 10,000 HN size is acceptable, then a smaller HN size would definitely work without any problems. SETI@Home (a well-known project in a distributed service domain) is a good example of distributed services that currently has 2.4 million connected nodes. Therefore, 10,000 nodes home network were chosen for our simulation. In addition, having 10 nodes in each HGw is one of our assumptions since there are many recent ubiquitous devices that have IP connectivity. For example, each home could have 4 mobile phones, 2 laptops, 2 tablets, and wireless sensor networks such as weather report sensors and so on.

We implement the weather report service sharing using Java language. The experiment is to be performed on three different architectures; RELOAD, Pure P2P and Centralized P2P. For each architecture, we count a number of signaling messages that used to delivery HS in four steps (i.e., Register, Publishing, Searching and Retrieving). We use only 1 registration step for each node since HGw, controller and sensors are usually fixed. We follow the details of routing in RELOAD as described by Bryan et al. [4]. The result according to Figure 7 shows that RELOAD has most registration latency than others in the Register step. This is because it requires forming an overlay network by pre-computing identities to find its position during Register step. However, this step is done less frequently and possibly only the time during any node's life. Next we will discuss the publishing step.

From Figure 7, there is no signaling message in the publishing step for Pure P2P since the nodes keep shared index to itself and the neighbor nodes have to flood a searching query. In RELOAD, we assume average route path to look up other node addresses and shared services using logarithmic number of nodes. For retrieving step, all architectures have the same signaling overhead. This is because the nodes are directly communicated after they already knew node destination. Overall, we can obviously see that Pure P2P has the highest signaling overhead while the centralized solution has the lowest signaling overhead among all architectures. Although RELOAD has more signaling overhead than Centralized P2P, it has several superior points. For example, RELOAD does not require any centralized server; each node has balanced loads; the setting avoids single point of failure, and bottle neck at a server. Most importantly, when comparing RELOAD with Pure P2P, the RELOAD solution has significantly less signaling overhead than Pure P2P.

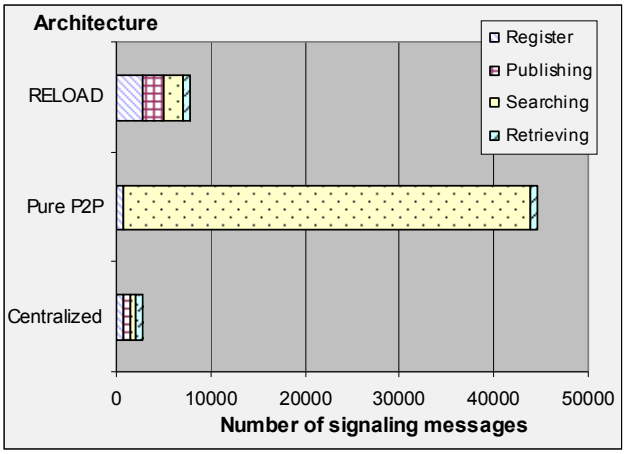

Figure 7. Number of signaling messages for four steps in different architectures.

Next we will compare the architecture in HS sharing context. From Table 1, Centralized architecture needs a central server that provides HS sharing index. This server requires a network administrator who is in charge of the server while Pure P2P and RELOAD require no administrator. Note that the symbol "+" in Table 1 refers to high performance service and that the architecture is capable of deliver service specification satisfactorily. The symbol "-" has the opposite meaning. The symbol "++" indicates a performance that is better than "+" in terms of service lookup speed and service lookup guarantee. Centralized architecture has the lookup cost of $O(1)$, received "++", while RELOAD has $O(\log N)$ where N is a number of nodes, received "+".

Table 1. A comparison of service specifications among three architectures.

Service Specifications	Centralized	Pure P2P	RELOAD
1. Maintenance Personnel	-	+	+
2. Failure Tolerance	-	+	+
3. Service Lookup Cost	+	-	+
4. Service Lookup Speed	++	-	+
5. Service Lookup Guarantee	++	-	+
6. Registration Latency	+	+	-
7. Single Point of Compromise	-	+	+

Regarding to the failure tolerance, Pure P2P and RELOAD have mechanisms to recover overlay network. Centralized architecture has the best service specification in satisfying service lookup cost while RELOAD gives quite a reasonable solution.

A single centralized server will be a single point of attacked target (also a single point of compromise) from intruders whereas Pure P2P and RELOAD prevent intruders from picking the attacked targets easily. Most essentially, RELOAD has several good specifications which are from a maintenance point of view, failure tolerance, and single point of compromise. It offers the acceptable service lookup comparing with Pure P2P. Note that the registration latency can be neglected since this step does not occur frequently as we have mentioned earlier.

It is interesting to note that in a large network, it would be useful to consider spatially close nodes only where RELOAD and Pure solutions should be implemented to provide more benefits in term of lookup cost since nodes are close to each other. The solutions can be used to reduce the communication cost better than Centralized solution (here the cost is fixed to $O(1)$). RELOAD and Pure solutions also offer more advantages in terms of maintenance personnel, failure tolerance and single point of compromise.

To conclude our chosen case study; the weather service sharing is an excellent example in showing that RELOAD is suitable to deploy for HS since the service lookup cost and speed are acceptable while we gain benefits of no maintenance personnel and high failure tolerance which are important in HS.

5. Conclusion and Future Work

In this paper, we introduced the concept of HS, and HS as a sharing service which will be a trend for the future services. We stated the issue of localize sharing HS in different topologies among three different network architectures. We proposed the delivery of HS using RELOAD architecture. Advantages and disadvantages were summarized. We found that HS sharing using RELOAD offers several benefits and is suitable for our case study that is the weather report service sharing. The reliability and efficiency are high while service cost and speed are reasonably acceptable.

In addition, the knowledge gained from our case study can be extended to other HS that has similar service requirements and service specifications. For further study we could consider other different aspects. Analyzing and evaluating these aspects should be done with the user security included authentication and the authorization in the decentralized way in order to thoroughly protect the service communities.

Acknowledgements

We wish to thank Beatrice Paillassa and Julien Fasson for their guidelines and ideas. Thanks to Faculty of Engineering, Prince of Songkla University for the financial support.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. parks, M. Handley and E. Schooler, SIP: Session Initiation Protocol, RFC 3261, IETF Network Working Group, 2002

[2] W. Werapun, J. Fasson and B.Paillassa, Network Architecture for home service delivery, *The fourth International Conference on mobile ubiquitous computing, system, services and technologies, UBICOMM 2010*, Italy, 2010

[3] W. Werapun, J. Fasson and B.Paillassa, Home Service Communities and Authentication, *Internal Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11*, China, 2011, 818-823

[4] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset and H. Schulzrinne, REsource LOcation And Discovery (RELOAD) Base Protocol, draft-ietf-p2psip-base-23, 2012

[5] D. Bryan, B. Lowekamp, and M. Zangrilli, The design of a versatile, secure P2PSIP communications architecture for the public Internet, *Parallel and Distributed Processing, IPDPS 2008*, Miami, 2008, 1-8

[6] Y. Yu, L. Chang and C. Shieh, Increasing service availability for resource constrained nodes in Peer-to-Peer communication systems, *Computer Symposium (ICS), 2010 International*, Taiwan, 2010, 789-792

[7] J. Maenpaa and J. Bolonio, Performance of REsource LOcation and Discovery (RELOAD) on mobile phone, *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, Sydney, 2010, 1-6

[8] M. Kolberg and J. Buford, Application layer multicast extensions to RELOAD, *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, Las Vegas, 2011, 1083-1087

[9] D. Bryan, P. Matthews, E. Shim, D. Willis, S. Dawkins, Concepts and Terminology for Peer to Peer SIP, draft-ietf-p2psip-concepts-04, 2011

[10] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan Chord: A scalable peer-to-peer lookup service for internet applications. Tech. Rep. TR-819, MIT LCS, March 2001, http://www.pdos.lcs.mit.edu/chord/papers/

[11] M. Jovanovic, F. Annexstein, and K. Berman: Modeling peer-to-peer network topologies through "small-world" models and power laws, IX Telecommunications Forum TELFOR 2001, Belgrade

[12] M. A. Jovanovic, Modeling large-scale peer-to-peer networks and a case study of Gnutella, MS. Thesis, University of Cincinnati, Cincinnati, Ohio, USA, 2001

[13] A. Medina, A. Lakhina, I. Matta, and J.Byers, BRITE: An Approach to Universal Topology Generation. *Proceedings of MASCOTS '01: The International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Cincinnati, Ohio, 2001