UNIVERSITATEA DIN CRAIOVA

ȘCOALA DOCTORALĂ DE FACULTATEA DE AUTOMATICA ȘI CALCULATOARE

SUMMARY OF DOCTORAL THESIS

IMPROVING LOOKUP EFFICIENCY IN OVERLAY P2P NETWORKS USING CHURN

CONDUCĂTOR ȘTIINȚIFIC:

PROF.DR.ING.COSTIN BĂDICĂ

DOCTORAND:

IYAS AL-ODAT

February 2015

Table of Contents

CHAPTER ONE: (INTRODUCTION) 1

OBJECTIVE AND MOTIVATION 1

CHAPTER TWO: (BACKGROUND PEER TO PEER NETWORK) 1

P2P DATA MANAGEMENT 1

P2P INFORMATION RETRIEVAL 2

CACH FOR WORLD WIDE WEB 2

DISTRIBUTED HASH TABLE 2

CHAPTER THREE: (P2P ARCHITECTURE NETWORK OVERLAY) 2

UNSTRUCTURE P2P OVERLAY NETWORKS 3

FLAT STRUCTURED P2P OVERLAY NETWORKS 3

HIERATICAL STRUCTURED OVERLAY NETWORKS 4

COMMON ISSUES 4

CHAPTER FOUR: (SURVEY OF P2P NETWORKS) 4

STRUCTURED NETWORK – CHORD 4

STRUCTURED NETWORK – KADEMLIA 7

UNSTRUCTURED NETWORK – GIA 8

CHAPTER FIVE: (SIMULATION ARCHITECTURE) 8

METHODOLOGIES 9

SIMULATION ARCHITECTURE 9

OVERSIM FRAMEWORK 9

INET FRAMEWORK 12

CHURN MODELLING 13

ROUTING METHODS 13

CHAPTER SIX: (PERFORMANCE EVALUATION) 14

SIMULATION SETUP 14

SIMULATION STUDIES AND RESULTS 15

PERFORMANCE P2P NETWORKS UNDER KBR 15

PERFORMANCE EVALUATION UNDER CHURN RATES 16

INVESTIGATION OF SCALABILITY OF P2P NETWORKS 16

OVERLAY LOOKUP INVESTIGATION 17

CONCLUTION AND CONTRIBUTION: 17

CHAPTER ONE: (INTRODUCTION)

We have seen an exponential growth of the amount of web content in the past twenty years ago since the beginning of the World Wide Web. As for now, recent studies report almost 30 billion web pages on the surface web a part of the indexed web by search engines, more than 500 million internet domains and about 1.5 billion Internet users. The size of web data has reached the order of petabytes and is constantly growing.

OBJECTIVE AND MOTIVATION

The academic community has implemented a number of DHTs as efficient, scalable, and robust P2P infrastructures.

DHTs make two assumptions which do not correspond to reality. These assumptions are as follows:

1- *Uniform communication:*
DHTs assume that communication between peers is uniform, that is, all peers have an equal probability of processing a request.

2- *Independence of the underlying physical network:*
DHTs have traditionally considered the physical IP-level network as a transparent layer, transmitting data from one point of the overlay to another. This assumption could make sense in small LAN where all users are close to each other and network delays are low.

CONTRIBUTION

Optimizing Lookup DHTs by design framework modules consist of three main models :1) The construction module. Responsible for constructing structured overlays from their flat peers. 2) The phenomenon. Joining and leaving peers in the network randomly is known as churn. 3) A concern which identified as crucial in P2P file lookups. Like the ability to analyse the details on the underlay network. We have chosen to study Overlay Network as follows: Chord and Kademlia, to represent the structured P2P networks. GIA, to represent unstructured P2P networks.

CHAPTER TWO: (BACKGROUND PEER TO PEER NETWORK)

Peer-to-peer (P2P) is a network where participating nodes share information equally, by using proper communication systems without necessarily needing central coordination. It is the opposite conception of client/server networks.

P2P DATA MANAGEMENT

To make Peer-to-Peer systems a viable architectural alternative for more technical and database-oriented applications than simple file sharing, support for powerful and expressive queries is

required. A couple of approaches have been suggested already on top of unstructured P2P systems.

P2P INFORMATION RETRIEVAL

P2P networks that distribute a global index over a large number of interconnected peers may be considered as a promising solution to cope with Web-scale retrieval document. While P2P networks consist of very large numbers of peers and indeed provide virtually unlimited storage capacities, there is no evidence about a real scalability of P2P Web search. Particularly, In the native use of structured or unstructured P2P networks for retrieval Web leads to unscalable network traffic and even for more sophisticated schemes, such as term to-peer indexing or hierarchical federated architectures, only little evidences of their scalability are available.

CACHE FOR WORLD WIDE WEB

Caching is a prevalent technique and used to improve the performance throughout centralized and distributed computer systems. A cache is simply a high-speed memory and used to store frequently accessed objects. For the World Wide Web, there are a number of different types of caches in the use of the internet. A browser cache typically uses a combination of RAM and local disk to store recently and used pages on behalf of one user. A proxy cache typically is a dedicated machine that caches pages on behalf of a group of users, and these users often share a connection to the internet. The nodes of a content distribution network are another type of caching, which is used on the Web today. Content distribution network caches (CDN caches) are similar to proxy caches in that they are dedicated machines that store frequently accessed Web pages. There are two key differences, and they are: CDN caches only store pages from those sites that pay for the content distribution service, and the mapping of users to CDN caches is dynamic rather than static.

DISTRIBUTED HASH TABLE

DHTs provide the same functionality of a traditional hash table — the standard Put (key, value) and Get (key) interface — but associating key-value mappings with participating nodes rather than hash buckets. Thus the location of an object is determined by the hash-value of its name. For instance, cryptographically secure hash-functions such as MD5 or SHA-1 map arbitrary strings to 128-bit or 160-bit hash-values, respectively. These functions can be used to map arbitrary object names into n-bit hash-values, where n depends upon the hash function which is being used. This defines an identifier space I = [0, $2n - 1$], where objects are assigned to. In a DHT, each participating computer is also assigned an identifier in the space I. At any instant, the current set of identifiers determines the set of partitions (hash buckets) that the hash-table has been divided into. Each host is the manager of a distinct partition (hash bucket) and being responsible for all objects whose names hash into that partition.

CHAPTER THREE: (P2P ARCHITECTURE NETWORK OVERLAY)

The Overlay is on the top of one or more underlying network. For Example, peer to peer network acts as an overlay network as a main existed network, which cannot provide the requirements of various applications. The node addresses come from that is not known before the routing message

and send to the logical address of the node. Some peers which are connected with each other and built on top of the internet are called an Overlay Network.

UNSTRUCTURE P2P OVERLAY NETWORKS

The overlay system does not have well cost scale because of flooding random, when searching about the query, then having to use complex search and they are not limited to index data in a network. The search of whole network grows as leaner with the network.

FLAT STRUCTURED P2P OVERLAY NETWORKS

Also known as: Distributed Hash Table (DHT), a flat structured network, down from a number of peers P that is together O. An important feature is that these networks enable a distributed hash table (DHT), which allows to uniformly to all participants in peer-to-peer system data.

In the following we have to see some kind of DHT.

Pastry

Is a generic, scalable and efficient substrate for peer-to-peer applications. Pastry nodes form a decentralized, self-organizing and fault-tolerant overlay network on the internet. Pastry provides efficient request routing, deterministic object location, and load balancing in an application-independent manner.

CHORD

Provides search function Lookup (key) denote for a set of IP, which is a 160-bit key in the map to set the IP addresses of the responsible nodes for this key. Each node has a 160-bit code, and Chord shows the ***s*** nodes whose identifiers immediately follow as a responsible key for this key, these are the main successor(s). To provide even a lookup search, half of the nodes is not in the 2-node network, the number of successors, ***s***, is 16 in the chord application.

TAPESTRY

Through the use of consistent hashing, it is possible to connect nodes, and leave the network with only minimal modifications. For a newly joining node n, a set of keys corresponding successor(s) to be assigned to N, and when n outputs, the reverse operation to be performed. Position and overlay routing infrastructure that provides location-independent route messages directly to the closest copy of an object or service by using only point-to-point links and without centralized resources. The routing and directory information within this infrastructure is purely soft state and easily repaired. Tapestry self-administering, fault tolerant, and resilient under load.

CAN

Content Addressable Network (CAN) decentralized P2P infrastructure provides hash table functionality in a web-like scale. To improve maybe will be the best examples are the introduction of the system for allocating peer-to-peer file. A node can maintain a routing table that the IP address and virtual coordinate zone of each of its neighbors is the holder.

HIERARCHICAL STRUCTURED OVERLAY NETWORKS

The superpeer-based model is attracted by the idea of hierarchical Routing in the internet. When we support, multiple domains, this model divide peers in different groups wherein each group may form its own structured overlay of: Chord, Tapestry, etc. each group, one or more of super peers participating in a super peer overlay, which super peers act as proxies of sending and receiving messages on behalf of colleagues in teams.

COMMON ISSUES

Bootstrapping

In connection with the peer-to-peer systems, which means that the process discovers one or more colleagues in peer which are a part of the system and help a new peer to join in the system overly network.

Churn

In the peer to peer network under churn (i.e. Randomly user arrived/ departure) flexibility with against node failure. On the structure peer to peer network is built upon of DHT, a user in DHT preserves two types of sets, and they are: routing table and successor.

CHAPTER FOUR: (SURVEY OF P2P NETWORKS)

In some of the cases has structured and unstructured P2P Networks. The first one (structure network) will include Chord, Kademlia, in the other one (unstructure network) represented by the GIA. The search for any file (look up) and architecture as we see about networks in some detail. Later we look at the performance evaluation P2P Network.

Almost of P2P network have a problem with Churn. Churn can describe it as a process are changes in height rate then it's the result from behaves of users that want to make the registration or un-registration of P2P network with or without cooperation from users in a network. In the worst case if the user not cooperation leaves the system, then the connection between neighbors maybe required for reconnection. This type of changes and variant of topology on the network makes it more randomly dynamic from the other type of network.

STRUCTURED NETWORK – CHORD

In the structure overlay network Chord it's implemented for make management and discover of resources in the network.

The Hashing in Chord is one of important adopted mechanisms; the hash will be at the IP address and produce the key this will attend with nodes. The hashing depends of load balance when all nodes receive roughly the same counts of keys, and maps of these keys that are responsible of nodes. In large network inefficiency node when try to keep all place of nodes in the routing table. The Chord has an ability to hashing by condition the node knows location a few of another node and the reason is distributing information a node own hash function that to arrive other nodes. Time consumes to keep our resolve message is O (log n) with other nodes if all N of nodes in the

network are steady state and with maintain are the same. The repeat procedure of behaviors nodes comings and goings from the network is maintained in routing. Each node should maintain his information's from time to time.

File lookups - Scalable key location

We can observe in Figure 1, the first entry in node 8's finger table points to node 14, because node 14 is the first node that succeeds [(8 + 20) mod 26]= 9. The last entry in the finger table points to node 42, because node 42 is the first node that succeeds [(8 + 25) mod 26]= 40.

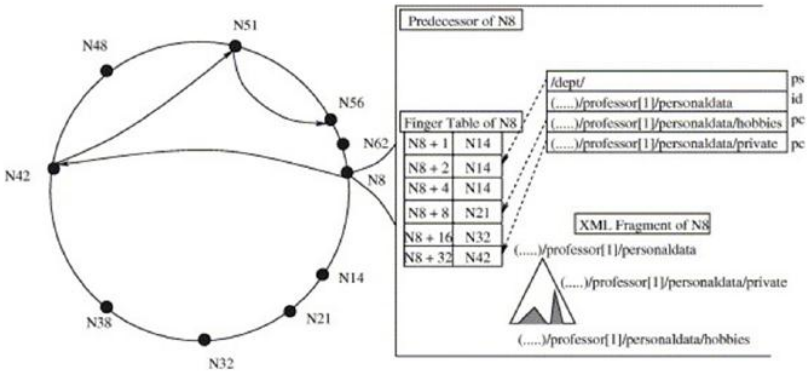

Fig1: Finger Table In Chord network

In Figure 2 there is of an exemplary P2P system 100 has ten traditional DHT nodes N1, N8 . . . N56. The DHT nodes N1, N8 . . . N56 each has a distinct DHT hash table 102.

The node N8 has hash table 102 Which stores references to, addresses or identifiers of various nodes N14, N21, N32 and N42 (as shown in column 104) that query for a limited number of ranges of key values 8+2O, 8+21, 8+22 . . . 8+25 (as shown in the column 106)

Node number N8 has a DHT hash table 102 is stored references to, addresses or identifiers of only a small number of the DHT nodes N14, N21, N32 and N42, and knows more about node N14 is closer to node 8 on the identifier circle than about nodes N21, N32 and N42 farther away.

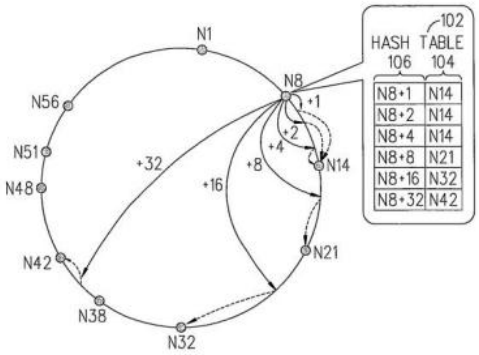

Fig 2: The Process File Look up.

In Figure 3 is P2P system 100 in node N8 has received a query 108 (lookup key 54) and uses the hash table 102 to determine that the query 108 needs to be forwarded to another node N42 so it gets closer to its final destination when is node 56. In this example, assume that node N8 Wants to find the successor of key 54. Since, the largest entry in the hash table 102 of the nodes N8 that precedes key 54 is node N42, then node 8 will ask node N42 to resolve the query 108. In turn, node N42 will determine the largest entry in its hash table that precedes key 54, i.e., node N51, node N42 will ask node 51 to resolve the query 108. Next, node N51 Will discover after checking its hash table that its own successor, node N56, succeeds key 54, and thus Will ask node N56 to resolve the query 108. At node N56 Will return its address 110, back to the originating node N8. As can be seen, this request routing involved several network hops from node N8 to nodes N42, N51 and N56.

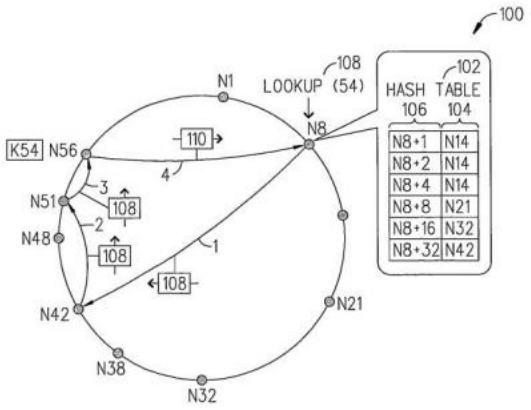

Fig 3: Lookup for key 54.

Dynamic operations and failures

An adaptation with the network when join the new node in the network or leaving the network normal or not normal. In Chord have mechanism to update the finger table for each node and also for the successor pointer from time to time this is by execute the stability of update, then the lookup can have a solution request as correctly.

We have three situations:

Situation number one: When kept the finger table in up to date, and the lookup is locate the correct successor with O(log n) time, then the performance it will be good and stabilization will be stop and lookup will be with a maintaining at O (log n).

Situation number two: The successor's point is in the correct way, but the entries in the finger table are not. Then we can see the correct look up and go on successful, but the difference is with the performance it maybe has some of the delay. We have to know that lookup still works because of not depend exactly where the query are in the network. The distances in the network are known by every node; every node knows the location of the other node. This is why performance has a delay.

Situation number three: the key is not yet copied at the new node or successor the point is incorrect. In this way the lookup has to be failed, and the application may be reworked again with lookup processing. Everything depends on the new node, when a new node happens to be between the predecessor and the target, then the node can stabilize by self with update finger table and to go to look up.

STRUCTURED NETWORK – KADEMLIA

It's one of the structure P2P networks using for improved efficiency to file lookup; the difference between Kademlia and Chord is how to calculate the distance between two nodes kadmelia he is using the XOR mechanism.

The lookup in Kademlia's has the same path of Chord this mean that chord with Kademlia have the same of uni-directional lookup, not important were initiated. The XOR metric when calculates the distance between two nodes, then XOR returns zero if two bits are equal and one if two bits are different.

When the Kadelima begin to build every node must to assign a global unique identifier it's the same when we call nodeID. Each node has pairs key and value, the value is stored file hashes or keywords. To ensure the uniform distribution of keys, a hash (160-bit SHA1 digit). Every node has to store in list of triples contain IP-address, UDP port and Node ID this information used for calculation of distance between list of buckets.

In the network size 2^3, then the maximum of node is eight in the network, as we see in Figure 4 seven of nodes are exist. At node number 5 we view the network of these moment node number 5 has three k-bucket, the bucket number one contain zero, one and two. Bucket number two contains three and four. Bucket number three contains six. When make XOR metric we can see it as: Distance between nodes six, one, five, and seven.

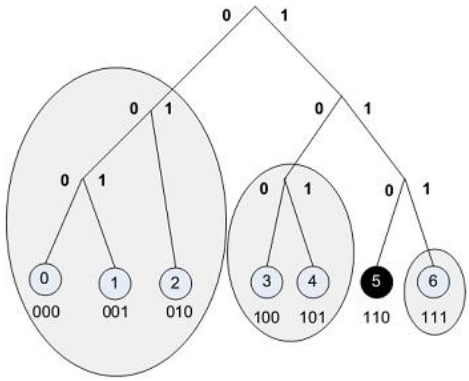

Fig 4: Kademlia network with 7 nodes and 3 buckets.

File Lookups

In chord and Kadelima is the same type of search to find the nearest target. When using the algorithm of recursive and asynchrony. That can be every peer in the kademlia network to choose node as same time to begin the tasks. When the pre-determined value α is three set by the

developer. When a new node found by some other RPC's that found it in past the source node send to RPC by recursively. If some of node doesn't have a respond during lookup within specific time for him that will make it unknown for future lookup until they are responding. During of looking up cycler if the results have not been seen that's closer and new in to target, then will make new RPC initiating another set RPC that is near and not have queried. From the bucket entire every nodes recipient receives are looking for to determine the request, the source node will be with under up-to-date again for known the nearest node for more answer in this process will be continuous repeatedly because every node catch to the nearest information surrounded of far node. This process will be stopped when new closer node is founded and bucket closer node own list closest to target key.

When $\alpha = 1$ this mean the bandwidth is the same way of Chord and the latency for determine that a node not with response. But the Kadelima is more efficiency in routing because can choose good node to route request.

UNSTRUCTURED NETWORK – GIA

Design – GIA: Four main components.

Dynamic topology adaptation: protocol that most nodes bring a short range high capacity nodes. The protocol adaptation ensures the proper connections (i.e. High degree) nodes, which received a lot of questions, in fact the ability to handle these issues.

Active flow control: scheme to avoid overloaded hotspots. This protocol for to realize clearness exists heterogeneity and some of adoptive with node available by flow control updates the status of high capacity nodes, and assigns tokens pending on available resources.

One-hop replication: The Nodes keep the pointers that contain some of the guides from the neighbors. The topology adaptation algorithm to ensure the compatibility between high capacity and high degree, the one-hop replica guarantees the higher capacity can be present answers for more numbers of queries.

Search protocol: based on partial random walks that sends queries to parts with high capacity nodes, which are usually in a better position to answer questions.

GIA clients keep a list of other clients in GIA network, a cache host which saves the high capacity nodes. The Cache is updated together with web-based caches hosting exchange data with their neighbors. The web-hosted cache will periodically review the list of high capacity peers and delete disconnected peers.

CHAPTER FIVE: (SIMULATION ARCHITECTURE)

The methodology which is used to evaluate performance of file lookup mechanisms in P2P networks and issues that is affecting the accuracy of our results.

METHODOLOGIES

P2P algorithms are often studied by using crawlers, emulators or simulators. In such crawler-based experiment, crawlers would be specially built peer nodes, which collect data from other nodes by visiting those nodes. By deploying multiple crawlers, a large proportion of a P2P network can be monitored. One of the drawbacks of this approach would be the fact that knowledge of a given P2P network would be limited to regions where the crawlers have been deployed. The other disadvantage of using crawlers is that users will have to be aware of how much overhead is produced while monitoring the actual P2P network experiment and will need to take this into account when interpreting the results.

SIMULATION ARCHITECTURE

The design and operation of the simulator, which contains works by example implemented, is a discrete event simulator and what behaviors are node simulation and how they are applied or churn can be specified. The better capture of the behavior of P2P users, simulated churn should be able to input parameters into probability distributions such as Weibull and Pareto. And also should be ability for extended when more easily implemented and P2P protocol, collect important data from the simulation.

What is the Omnet++

It's an Objective Modular Network Testbed in C++. Designed to simulate computer networks, multi-processors and other distributed systems. Its applications can be extended for modelling other systems as well. It has become a popular network simulation tool.

User-interfaces OMNET++ SIMULATOR

The Omnet++ user interface is used with the simulation execution. The Omnet++'s design allows the inside of the model to be seen by the user, also allows the user to initiate and terminate simulations, as well as change variable inside simulation models. These features are handy during the development and debugging phase of modules in a project. The graphical interface is a user friendly option in Omnet++ allows access to the internal workings of the model.

Tkenv User Interface Tkenv: is a portable graphical windowing user interface. Tracing, debugging, and simulation execution are supported by Tkenv. It has the ability to provide a detailed picture of the state of the simulation at any point during the execution. This feature makes Tkenv a good candidate in the development stage of a simulation or for presentations. **Cmdenv**: is designed primarily for batch execution. It is a portable and small command line interface that is fast. It compiles and runs on all platforms. Cmdenv uses simply executes all simulation runs that are described in the configuration file.

OVERSIM FRAMEWORK

OverSim provides several common functions for structured peer-to-peer networks to facilitate the implementation of additional protocols and to make them more comparable. OverSim utilizes the graphical interface of OMNeT++ to display overlay and underlay topology, thus allows for intuitive debugging. Additionally, the framework provides a central module for gathering and

processing statistics. Oversim's postprocessing scripts facilitate the generation of publication quality plots. The oversim's modular architecture allows the modelling of all components of a P2P network all Modules can be easily exchanged or extended. The resulting flexibility facilitates code reuse. Several exchangeable underlay network models allow simulating complex, heterogeneous underlay networks as well as simplify networks for large-scale simulations. With OverSim simulations of overlay networks with up to 100,000 nodes are feasible.

OverSim Models

Simple: The Simple Underlay is the default underlay model for OverSim. It combines a low computational overhead with high accuracy, making it a good model for simulating large overlay networks. In this model, nodes are placed into a n-dimensional Euclidean space. The delay between two nodes is then calculated by the distance between those nodes. Additionally, each node is assigned to a logical access network characterized by bandwidth, access delay, jitter and packet loss parameters allow the simulation of heterogeneous access networks. Node mobility can be achieved by changing coordinates, access network characteristics and the IP address of a node. To model bandwidth effects, each node contains a logical sending queue. The Simple Underlay allows the simulation of underlay network partitioning and merging. **INET:** For simulations of heterogeneous access networks, backbone routers and terminal mobility, OverSim provides an underlay model based on the INET framework. Here, the IP stack is completely modelled and even routers can be part of the simulated overlay. INET also contains several MAC protocol implementations, which e.g. allow to model wireless IEEE 802.11 interfaces. **SingleHost:** The SingleHost Underlay provides real network support for OverSim. It acts as a middleware to support deploying overlay protocols developed for OverSim on real networks.

Architecture Of Oversim Framework

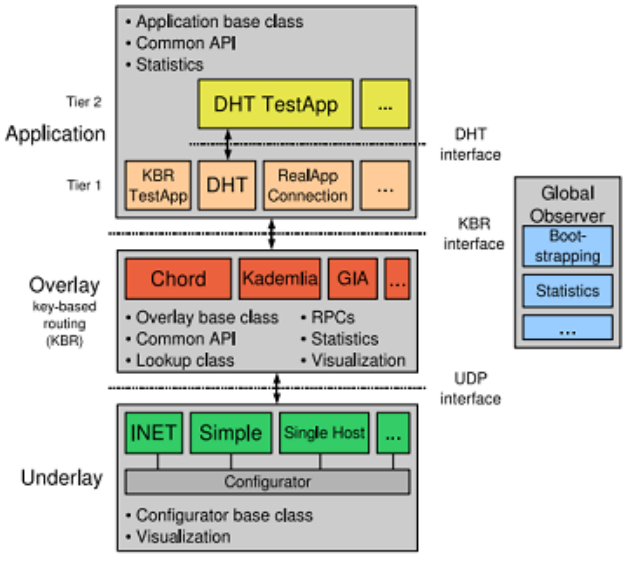

Fig 5: Modular of OverSim Architecture

Underlay Network

Simple underlay: data packets are sent directly from one overlay node to another by using a global routing table. Packets between overlay nodes get delayed either by a constant period of time or for more realistic scenarios by a delay calculated from the node distance. **SingleHost underlay:** OverSim instance, only emulates a single host, which can be connected to other instances over existing networks like the Internet. To enable the emulated host to communicate over real networks, can be implemented a new event scheduler for OMNeT++. In contrast to the default scheduler, which aims for fastest possible simulation, this scheduler slows the simulation speed down in real time. **The INET underlay model:** derived from the INET framework of OMNeT++, which includes simulation models of all network layers from the MAC layer onwards. It is useful for simulations of complete backbone structures. Since the INET framework is not optimized for large scale networks, we had to profile the code and added hash table based routing and interface caches for faster forwarding of data packets.

Overlay protocols

Several functions that many overlay protocol implementations have in common and integrated them into simulation framework. *Overlay message handling (RPC and statistical data).* Simplified dealing with timeouts and packet re-transmission due to packet losses. It also collects message related statistical data like the number of sent, received, forwarded and dropped packets per node, as well as the ratio of data to signalling traffic. *Generic lookup function.* Separates the common functionality of lookup mechanisms in structured overlay networks and provides a generic iterative and recursive lookup interface. This way, for each overlay protocol only a method to query the local routing table has to be implemented, which returns the closest node in the overlay topology. The lookup function also includes basic support to simulate the malicious node behavior. When the overlay structure against attacks of malicious nodes largely depends on the number of disjoint overlay paths that exist between two arbitrary nodes in this mechanism make redundant paths and can be used to measure the effects of some attacks on lookup success. *Support for visualization of the overlay topology.* The user interface of OMNeT++ supports debugging of new overlay protocols by showing transferred messages in detail. It is also possible to watch and even change overlay specific routing tables and other internal state during running. *Bootstrapping support.*

Applications

The communication between overlay and application makes us to use the Common API. Every overlay protocol, which wants to use this API has to provide at least a key-based routing interface (KBR) to the application. With the Common API design a wide range of different applications that rely on, key-based routing can be valuated with exchangeable overlays. Currently there is a KBR test application implemented, that depending on parameter values periodically sends test messages to random overlay keys or node IDs and records message delay and hop-count. More complex applications can be composed of several logical tiers.

KEY-BASED ROUTING API

A key is a 160-bit string. A node handle encapsulates the transport address and nodded of a node in the system. The nodeId is the type of key; the transport address also, for example, an IP address and port. Messages (type msg) contain application data of arbitrary length.

Routing messages

void route(key K, msg M, no de-handle hint). This function is forwards a message, M, towards the root of key K. In argument hint species a node that should be used as a first hop in routing the message.

void forward(key K, msg M, nodehandle nextHopNode). This forward function is invoked at each node that forwards message M, including the source node, and the keys root node.

void deliver(key K, msg M). This function is invoked on the node that is the root for key K upon the arrival of the message M. The deliver upcall is provided as a convenience for applications.

Routing State Access

nodehandle [ ] local lookup(key K, int num,boolean safe). call produces a list of nodes that can be used as next hops on a route towards key K, such that the resulting route satisfies the overlay protocols bounds on the number of hops taken.

nodehandle [ ] neighborSet(int num). Unordered list of nodehandles that are neighbors of the local node in the ID space. Up to num node handles are returned.

nodehandle [ ] replicaSet(key k, int max rank). Returns an ordered set of node handles in which replicas of the object with key k can be stored. The call returns nodes with a rank up to and including max rank.

update(nodehandle n, bool joined). Invoked to inform the application that node has either joined or left the neighbor set of the local node as that set would be returned by the neighborSet call.

boolean range (nodehandle N, rank r, key lkey, key rkey). Provides information about ranges of keys for which the node N is currently a re-root. The operations returns false if the range could not be determined, true is determined.

INET FRAMEWORK

Contains IPv4, IPv6, TCP, SCTP, UDP protocol implementations, and several application models. The framework also includes an MPLS model with RSVP-TE and LDP is signalling. Link-layer models are PPP, Ethernet and 802.11.

Communication between protocol layers

In the INET Framework, when an upper-layer protocol wants to send a data packet over a lower-layer protocol, the upper-layer module just sends the message object representing the packet to

the lower-layer module, which will in turn encapsulate it and send it. The reverse process takes place when a lower layer protocol receives a packet and sends it up after DE capsulation.

Multiprotocol Label Switching – MPLS. Is a scalable, protocol-independent transport. In an MPLS network, data packets are assigned labels. Packet-forwarding decisions are made solely on the contents of this label, without the need to examine the packet itself. The primary benefit is to eliminate dependence on a particular OSI model data link layer technology.

Resource Reservation Protocol-Traffic Engineering -(RSVP-TE). It supports the reservation of resources across an IP network. Applications running on IP end systems can use RSVP to indicate to other nodes the nature of the packet streams they want to receive.

Label Distribution Protocol – LDP. Defined for distributing labels. It is the set of procedures and messages by which Label Switched Routers (LSRs) establish Label Switched Paths (LSPs) through a network by mapping network-layer routing information directly to the data - link layer switched paths.

CHURN MODELLING

One of the major problems suffered by P2P networks is churn. Churn is a term referring to random disruptions of connections between peers. Churn has been categorized into environmental and characteristic factors. In the environmental factors the parameters concerned are a number of active peers and theirs joining frequency. With characteristic factors, the parameters concerned are the re-transmission times of query failures.

NoChurn. Nodes will be added until targetOverlayTerminalNum is reached; after that, the network will remain static. Parameters: None. **LifetimeChurn.** On creation of a node, his lifetime will be drawn randomly from a given probability function. When this time is reached, the node is removed. A new node will be created after a dead time drawn from the same probability function. Parameters: lifetimeMean: The mean lifetime (in sec). lifetimeDistName: The function used for drawing the lifetimes (default: weibull). **ParetoChurn.** Node behaviour: Similar to LifetimeChurn, a node's lifetime is drawn on creation. However, this is done in a two-stepped process. Parameters: lifetimeMean: The mean lifetime (in sec). deadtimeMean: The mean deadtime (in sec). Notes: This churn generator results in heavy tailed session times similar to empirical results from P2P file sharing networks. **RandomChurn.** Node behavior: At fixed intervals a random number is drawn. Depending on this number, a random node is added, deleted or migrated.Parameters: targetMobilityDelay: Timespan in seconds between two actions. creationProbability, migrationProbability, removalProbability: Changes the probability of the actions taken when a number is drawn.

ROUTING METHODS

Description of the different Key Base Routing - KBR types: All implemented KBR protocols that make proper use of BaseOverlay support the following routing modes, which are selected by the .routingType parameter: **Iterative:** The originator starts a lookup by sending one (or multiple) FindNode RPCs to the closest next hop stored in the local routing table. The received FindNodeResponse contains better next hop nodes, which are successively queried by the

originator until a FindNodeResponse with isSibling=true is received. Finally the message is directly sent to this sibling (= ”closest node”). **Exhaustive-iterative:** This is similar to the iterative case above, but the isSibling bit is ignored. The originator instead continues to query all closest nodes, until no better nodes are returned. This increases bandwidth consumption and lookup latency, but is more secure. **Semi-recursive:** The originator encapsulates the message in a BaseRouteMessage and forwards it to the closest next hop stored in the local routing table. The message gets forwarded recursively until the sibling node (= ”closest node”) for the destination key is reached. An optional response to the message is sent directly to the originator. **Full-recursive:** Similar to semi-recursive routing, but the response is routed recursively back to the originator Source-routing-recursive: Similar to semi-recursive routing, but the response is routed back to the originating along the reverse path of the routed message.

CHAPTER SIX: (PERFORMANCE EVALUATION)

We present results of the performance evaluation study of Chord, Kademlia and GIA, and their lookup mechanisms in particular. The aim is to investigate the effects of churn on the performance of these P2P networks. The results were obtained by means of stochastic discrete-event simulation, using OverSim as a simulation tool.

SIMULATION SETUP

We specify the average activity period that gives us best reflect about reality user behavior. We suggested that a period of activity from some of minute to one hour. We suggest also that the average activity period time of a peer be 600, 1200, 1800, 2400, 3000 and 3600 seconds.

The performance metrics to represent the performance of file lookups processes:

Hop-count: Is a basic measurement of distance in the Network, the distance between the source node started the lookup and looking up for the destination node (the position of the target value). **Bandwidth-consumption:** includes the amount of traffic generated by the routing table updates and file searches, measured by the number of messages generated for a certain period. Latency: specified the time of the stabilization process from the keep of Churn, moreover the duration that time for the resolve lookup file process from the Initialize until responded this measure as milliseconds. Delivery ratio in Chord and Kademlia: ratio of successful deliveries of file lookups to the destination node over a given time interval. **Satisfaction-Level-In-GIA:** Represents the level at which a peer GIA is satisfied with the current set of neighbors for solving file searches. Satisfaction varies a value between 0 and 1, where 0 is dissatisfied and satisfied one. This value is measured by the quality of the peer receives answers from all the neighbors for resolving file searches.

Simulation Models

The simulation exists some script file for beginning the simulation by users need to specify their experiment in the omnetpp.ini script file, on the other side P2P network properties and the lookup parameter exist in the default.ini script file. If the user chooses to Initialize with a default value, then don’t need to explain in omnetpp.ini otherwise when their own value they must explain in

omnetpp.ini. In the simulation all parameters are defined in omnetpp.ini or default.ini that is input in Oversim when the oversim is an overlay simulator used the OMNET++ in the background to handle the detail in the underlay.

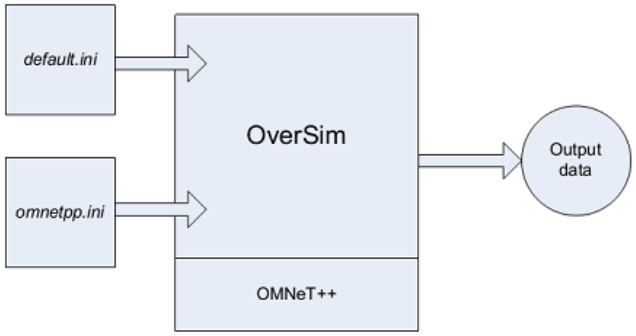

Fig 6: Simulation architecture.

In Figure 6. Simulation parameters are defined in the default.ini and omnetpp.ini script files, which are the inputs to OverSim. OverSim is an overlay simulator, and it makes use of OMNeT++ in the background to handle simulation details in the underlay. Output data is prepared by OverSim. In default.ini file contains all the default values, and will remain unchanged during simulation unless otherwise overwritten in omnetpp.ini. The omnetpp.ini holds simulation scripts, where users can define their specific experiments.

SIMULATION STUDIES AND RESULTS

The effect of churn in GIA, Chord and Kademlia was considered to compare between to gather. We will see the four effects:

PERFORMANCE P2P NETWORKS UNDER KBR

Comparisons between the efficiency of lookup file and rate that be successful under routing iterative and recursive in Chord and Kademlia. The full capacity of the network has when reached at 500 peers. Hop count: is a basic measure of distance in the Network, the distance between the source node started the lookup and looking up for the destination node. Bandwidth. Using capacity of networks.

Hope Count. In both of the methods are not most affected with churn because there are fixed throughout different average activity period. Generally, the recursive was able to resolve lookup file at fewer hop counts. Recursive routing Kademlia illustrate the best performance in the latency of average at 600 seconds of the average activity period, which is around 0.05 milliseconds.

Latency. They achieved a small latency in recursive routine, and the all result are fixed at different whole period active. Recursive routing results showed about 1% improvement in Chord and about the 2% improvement in Kademlia. As we known about XOR-based metric topology that used in Kademlia that permit to get lookup file processing from the same number of neighbors that stores in Kademlia routing table.

Bandwidth Consumption. In Chord performed effectively using up about 30,000 messages per simulation of six hours with recursive routing, while the iterative routing consumes about 50,000 messages per simulation of six hours. In Kademlia the iterative routing effectively lower churn rate, the consumption of high bandwidth churn rate is increasing by 80%. On the other method of recursive routing method in Kademlia is present the lower consumption of bandwidth. In Kadmelia can add the message with file lookup to see bandwidth and maintenance of the network when we compare with Chord network.

Deliver Ratio. In Structure networks has same efficiency when there keep for all the received in different churn rates. But in recursive routing method we can observe that is more churn rates to deliver period and with direction, decreasing when we looking at the numbers as separated between both of routing.

PERFORMANCE EVALUATION UNDER CHURN RATES

We applied average activity period in 600, 1200, 1800, 2400, 3000, 3600 seconds in different 500 peers.

Hop Count. In Chord had an average of 4.5 hops; Kademlia had an average of 5.5 hops and GIA at around 1.0 hops. The node can respond when received the query not just only matches from contents also can respond with matches from content by all neighbors. The GIA had better results because GIA networks use ”one hop” structure, each node maintains a list of keys (files) from of its neighbors. We can see that the structure and unstructured network can resolve file lookups under affection of churn.

Bandwidth. In GIA with decreasing trend and the average period are increasing. In GIA depend on high capacity to resolve the file lookup because the peer can keep more of a list of keys from the neighbor.

Latency. In both structure networks is smaller than GIA. The plot the varied network structure is between 0.1 and 0.6 milliseconds of latency. We can see that the GIA network appears to increase the linear and with decreasing of churn rate, from around 1.4 milliseconds until 5.5 at from period time 500 until 3600 seconds of the average activity period. The entire lookup file, send to the node with the highest capacity level in GIA network, this is a reason for high delay when replies.

INVESTIGATION OF SCALABILITY OF P2P NETWORKS

We increased the number of nodes in the network from 500 nodes to 1000 nodes that’s mean we grow the size of the network.

Hop count. In Chord has an average of 5.8 hops. Kademlia has an average of 5 hops. GIA performs with around 1.4 hops. When comparing the result with (Performance evaluation), average hop counts in Chord and GIA increase by about 1.0 %, while in Kademlia the average hop counts increase by about 0.5 %. All three networks seem to fit well enough network size grows.

Latency. The nodes in structured networks share equal amounts of responsibility for solving look up for files, the accumulated delays are very small. On the other hand, the GIA network do more

better performance in large network compare with small network as the churn rate falls below the value corresponding to 1.4 seconds of the average active period of time, the greater average network latency grows faster. In the cases examined the network delays GIA series between 1 and 2.5 seconds. When the size of the network is 100 peer the latency of GIA is between 0.01 to 0.02 seconds, the reason for resolve with larger network that more peers connected with high capacity node and this is a chance to be in higher compare at small network where the group is smaller than the processing lookup file likely for takes less time.

Satisfaction Level. In GIA the satisfaction level increasing with less dynamic network, in this way the large network have a better average satisfaction level when compare it with small network the range in a large network is between 1.6 and 1.8 but at small network is between 0.8 and 0.9. This is confirmation for large network with more peers have a more neighbor be the satisfaction level to be higher.

OVERLAY LOOKUP INVESTIGATION

Stabilization process at the high capacity arrive to 100 peers. We assume that another peer joined in a network, then will specify 100 random queries.

Responds Messages. In GIA network have decreasing of responding over time is increased from 0.008 to 0.002. In the structure network, we can see that both of them have constantly responded during lookup messages in the different simulation time.

Lookup Failed. In Chord Get failed is increasing during an increasing time of the lookup as linear, but in the other hand Put failed is not possible. In KAD Get and put under network is not possible to be failed because the KAD network to make churn have to re-build the network from bootstraps.

Lookup latency. In Kademlia PUT and GET have a less Latency than Chord PUT and GET as we can observe too the Latency for PUT and GET is approximately is the same this is because the bandwidth is fixed.

GET Ratio. A successful GET request for a key is a request which delivers the value previously stored under that key. The GET success ratio is the number of successful requests divided by the number of all GET requests. A success rate is 1 in Kademlia because of network errors such as timeouts or packet loss or the effect of members leaving the system. The rate is a metric for the reliability of the system, implementations values need to be republished periodically otherwise they are dropped by the storing nodes. The lifetime and the republish interval of the values were chosen to be infinite to exclude side effects from the measurement.

CONCLUSIONS:

In recent years, volumes of P2P significant increased, traffic in all parts of the world for internet service providers (ISPs) main varieties for make this network accepted. P2P network is no need for a central server's, then direct connection between clients are allowed and size of network is growing rapidly. For preserve network from deterioration we need good plan management and

effective network algorithms to discover of resources under network, also preserve other services from deteriorated.

In Chapter 4, we have presented the architecture of Chord, Kademlia and GIA, in this overlay have outlined networks, management schemes and resource discovery phases of each of these networks. Unlike of these networks to be as traditional networks like server and client network. Peers can join and leave a network unexpectedly, and the network becomes unstable. Without a central management entity, individual peers are responsible for updating their own routing table and resolving file lookups between each others. During the procedure of file lookup, peers will depend on each others when searching for a match for data requests. Because the resources are limited in most of peers, an efficient routing scheme is definitively. In a large dynamic network if efficient mechanisms are not implemented, these issues can be climbed to halt operations in a short period of time. Chord and Kademlia are categorized as structured networks that have adopted DHTs. In structured networks, there is a set of guidelines which each peer will need to follow to connect to the network, manage its own routing table and handle file lookups. GIA is categorized as a hybrid network under unstructured networks. The characteristics of GIA are that all peers will connect to high capacity nodes and route all file lookups to the high capacity nodes. High capacity nodes will have lists of resource points from nodes that are connected to them. In Chapter 5, we discussed our experiment methodology, surveyed P2P simulators, and considered models of churn, as a vital characteristic of P2P processes.

In Chapter 6, we evaluated Chord, Kademlia and GIA through simulation studies.
Our result indicates that p2p network under the key base routing method in both types of the structure network (chord and kademlia) they have best of consume smaller bandwidth and smaller delay under recursive routing method. But we can see the deliver ratio of messages are unsatisfactory. Also, when applied average activity period at 500 peers for measurement performance evaluation under churn rates we observe that GIA resolve file lookups, with better result and consume smaller bandwidth, but we observe that GIA suffer in high latency compare with structure network (chord and kademlia). When we grow the size of network to investigate of scalability in p2p network, we conclude that GIA large network performed better than smaller network (latency is performing better), also have a high satisfaction level after the increase size of the network. In the final test we have made a lookup investigation, we observe that GIA have increase respond messages when the size of the network are increased. On the other hand, the structure network has a constant response message during times for file lookup. We can observe also under structure network failed lookup in kadmelia is not possible, but in Chord GET failed are increased during increasing time as a linear. Also, we can see lookup latency in kadmelia PUT and GET has less latency than Chord.