Elucidating the Dynamics of Peer-to-Peer Hypercube Structure

Srinivas Kasturi and Maitreyee Sapatnekar, University of Winnipeg, Winnipeg, Manitoba, Canada, R3B 2E9

***Abstract*—A major challenge in any peer-to-peer network is to effectively provide services and be functional despite an interruption. Resilience against failure and the capability to self-organize and transfer data efficiently are some of the key factors that make our algorithm more attractive. This paper provides a unique theoretical framing of peer-to-peer architecture, a methodology for analyzing scalability and reliability, and a useful means for understanding the hypercube dynamics. A relatively simple and scalable hypercube topology is employed to represent and evaluate these interactive dynamics.**

***Keywords*- Peer-to-Peer Systems, Hypercube, Scalability, Reliability**

I. INTRODUCTION

PEER-TO-PEER architecture, in recent years, has evolved dramatically and a new wave of collaborative peer-to-peer systems has emerged, proving to be extremely efficient in its ability to function, store and handle data, scale and adapt in the event of a system failure with high fluctuations in the population of nodes. These architectures possess the potential to fasten communication between the nodes without the need of a centralized server approach. The management, maintenance and operational dynamics is divided into tasks assigned to the users. Here, with the use of theory and experiments on peer-to-peer architecture, it can be demonstrated that, for a wide range of conditions, the performance and reliability quotient of the system can be examined.

A. Design of P2P Framework

Peer-to-peer system design focuses primarily on exchange of resources between peers. It is an extension of the distributed system architecture, where the workload is divided between potent participating peers who have identical liberties and capabilities. There is an underlying assumption that peers generously contribute resources to the common pool and engage in data sharing activities in many peer-to-peer systems.[1] Unlike the traditional client-server approach, the peers play the role of both: consumers and suppliers.

With the difference in whether or not a strict structure is imposed on the overlay network, the peer-to-peer networks are mainly classified as Structured and Unstructured peer-to-peer networks. Gnutella is one such example of unstructured P2P protocol facing the limitations in scalability.[2] A distributed hash table(DHT) is commonly implemented in structured P2P networks with BitTorrent being a notable example of this approach. For the purpose of this paper, we have studied the peer-to-peer design framework to accommodate node creation, file generation and retrieval and introduction of an adversary in order to analyze the performance metrics.

B. Hypercube

Hypercube topology, which belongs to the popular families of network topology, is studied and implemented for the purpose of this paper. Peers are organized in the system in an n-dimensional hypercube.[7] As in any other structured P2P network, a distributed hash table(DHT) is constructed for effective routing between peers in the n-dimensional hypercube.

II. RELATED WORK

Extensive research has been carried out recently in the area of peer-to-peer computing. One such paper attempts to put some light on the common problem of lowering unnecessary message transmission which in turn helps in reducing overall bandwidth consumption in a Peer-to-Peer(P2P) network.[3] The core idea is to compute the distances between nodes to achieve query optimization by adopting a labeling scheme. The need to find the distance between two nodes by sending out queries is thus eliminated.

Discussion and research is carried out in the paper by Mario et al., regarding scalability problems, restricting the count of nodes in the system, network overload issue and longer search times in unorganized P2P networks.[4] These issues are resolved by implementing an algorithm based on a graph topology which supports efficient broadcast and search. This makes it possible to establish communication with all the nodes with the least amount of message transmission. Proficiency of this scheme is achieved by utilizing a global ontology which helps in efficient concept-based search.

Another paper by Ghodsi. A, related to this field of study, puts forth an argument on the suitability of structured overlay networks in carrying out group communication efficiently.[5] Here, an algorithm based on Application Level Multicast(ALM) approach is utilized which aims at eliminating redundant messages in the broadcast method.

A framework for examining peer-to-peer content distribution is studied extensively in the paper by Stephanos et al., in terms of how the characteristics such as security, scalability, performance and resource management are affected by current peer-to-peer system architecture design limitations.[6]

In contrast to the above papers, this paper touches on and combines various theoretical concepts pertaining to the area of P2P systems such as node creation, file generation and storage, file retrieval and insertion of an adversary using multithreading.

III. Methodology

The objective of this paper is to employ and exploit the concept of multithreading in order to achieve the desired results. A structured P2P network is constructed based on the hypercube topology consisting of n dimensions. The number of dimensions of the hypercube helps in determining the total number of nodes in the hypercube structure which is 2^n. Each node has its own thread which supports file generation and retrieval. The reliability of the system is also analyzed by introducing an adversary which attempts to destroy the nodes/peers by killing the threads during their execution. The system architecture employed in this paper follows a certain process flow which is illustrated in Figure 1.

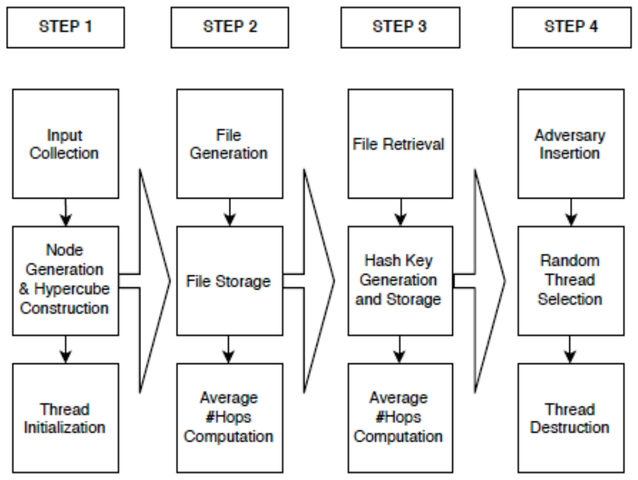


Fig. 1. Process flow diagram of Peer-to-Peer hypercube system. This illustration explains the steps involved in the design and management of the system.

A. Node Generation and Thread Initialization

This section explicates the organization of a structured P2P system as a n dimensional hypercube. The value of the dimension variable n plays a key role in indicating the number of nodes/peers that the hypercube architecture can support. The maximum number of nodes which can exist in the hypercube of dimension n is computated as 2^n. The virtual machine used for the purpose of the experiments in this study have the limitation of supporting only a particular number of dimensions of the hypercube. This limit of n is derived with the help of hypothesis and experiments.

Subsequently, threads are spawned and simulated for each node in the system. The concept of threads is adopted for the purpose of this study because they possess the capability to support and manage multiple operations concurrently with high efficiency. The threads are thus created and each one of them is assigned with a different binary identifier such as 00, 01, 10 and so on. This step deals primarily with the thread initialization and establishing a connection between the generated nodes.

B. File Generation and Storage

Data storage is the next goal in our study after the successful creation of nodes and thread spawning. Generation of physical files by each node is carried out in this step. Each file will consist of random 4-character strings comprising of only uppercase ascii characters and digits representing an encrypted file produced by the user. A hash function is employed in this scenario in order to compute the key for the encrypted file by adding the binary representation of the 4 characters in the string and by applying a modulo operator (%) with divisor being 2^n. The file containing the 4-character string will thus be stored in the node having an identifier equal to the computed hash key for that particular file. Production of data files by each P2P node follows a certain process which starts by initializing the number of files to 0 followed by producing a random file and its key k. Finally, the file is stored in the node with identifier identical to key k. The maximum number of files that a certain node can produce is restricted by the system and this number is represented by m which is derived by hypothesis and experiments. The average number of hops required for storing the data in the system for each m, n(where n varies from 2 to the maximum value it can take) is then calculated to examine the performance of the system.

C. File Retrieval

In this section, the focus now shifts from the node creation, thread initialization and file generation and storage to data lookup and retrieval in the structured P2P system. Similar to the file generation and storage, each node will produce the data files but with the exception of storing the keys of the generated files. File retrieval will be carried out based on the value of each file key saved while producing the files. The maximum number of files that a certain node can retrieve is represented by p which might be a value between 1 and m.
When a certain file needs to be retrieved, its key is chosen from a pool of m keys which are previously stored in the system. Once the key matching is done with the respective identifier of a node, the file retrieval succeeds. The average number of hops required for retrieving the data in the system for each m, n, p(where n varies from 2 to the maximum value it can take, m varies from 1 to its derived maximum value and p varies from 1 to the current value of m) is then calculated to analyze the system efficiency.

D. Introducing an Adversary

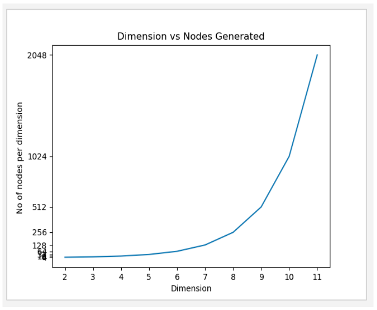
Reliability is an important factor in a structured P2P system and needs to be examined in order to ensure system tolerance and performance in the event of a failure. The reliability of the system proposed in this paper is tested by introducing an adversary which is merely an evil process or a thread gone rogue. This evil process aims at destroying peers randomly in the P2P system by killing their respective threads.[8] The maximum number of nodes attacked is represented by q which might be a value between 1 and n. The amount of time the adversary takes to plan its attack is represented by r. The adversary chooses a random node out of n existing peers and kills the thread of the node having the respective identifier. The evil thread or adversary executes simultaneously with the execution of file retrieval activities. This helps in analyzing the effect of the adversary on the performance and reliability of the retrieval process and the system in general.

IV. Experiments

In order to study the performance metrics of the complex peer-to-peer system, the following experiments were performed.

Goal 1

By changing different initial conditions, we observe that the number of dimensions of the hypercube given by *n* is directly proportional to the number of files generated by a particular node given by *m.* Based on the code execution and experiments, the maximum value that *n* can take is 11 having the maximum capacity to create 2048 threads in total. If the value of *n* is given as 12, the system can still create maximum 4093 threads out of 4096(the remaining three threads are used by the core system) but does not support the hypercube functionality and throws an error. The relationship between the *n*(number of dimensions of the hypercube and the total number of nodes generated is illustrated in Figure 2.

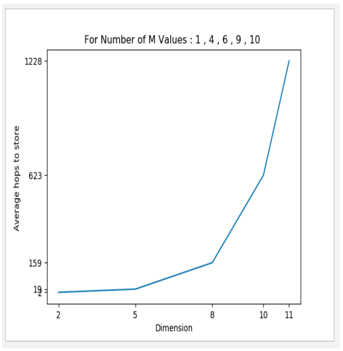


No of dimensions	Nodes generated
9	512
10	1024
11	2048
12	Crashed

Fig. 2. Shows the relationship between n and the number of nodes generated in the system

Goal 2

The experimental results demonstrate how the nodes generate files and store data in the hypercube. The files are generated using threading.local and a local phyical file is created for each thread where it can store the 4-character string(encrypted file). The threading.enumerate function is used if the thread wants to store data in another thread. Figure 3 describes how n is directly proportional to the average number of hops to store denoted by *ahs* and is computed for increasing m values. If we consider n as 8, then the average number of hops is around 169 for various m values. The table in Figure 3 comprises of 5 set of values which are used for the experiments and graph plotting.

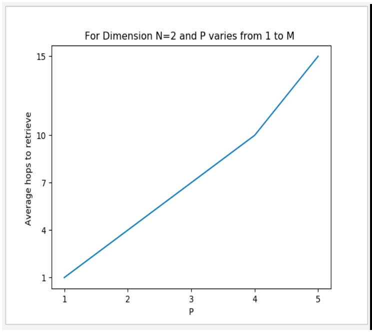


#runs	*n*	*m*	*ahs*
run1	2	1	2
run2	5	1-50	19
run3	8	1-20	159
run4	10	1-7	628
run5	11	1-4	1228

Fig. 3. Illustrates the relationship between *n* and the average number of hops *ahs* with varying *m* values.

Goal 3

From the experimental observations, it can be derived that p is the parameter which is in the range of 1 to m which is utilized to retrieve the files generated from the respective nodes in m iterations. In order to maintain a stable state of the system, files which are retrieved are within the range of n = 8 and m varies between 1 to 7 and p varies between 1 to 7. If an attempt is made beyond these values, the time required by the system to perform the computations increases considerably having high CPU utilization and thus is not able to retrieve any files or calculate the number of hops. Figure 4 explains the relationship between *p* and the average number of hops to retrieve denoted by *ahr*, required to retrieve files. The table in Figure 4 comprises of 5 set of values which are used for the experiments and graph plotting.

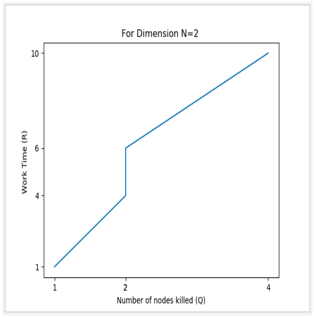


n	*m*	*p*	*ahr*
2	1	1	1
5	1-4	1-4	12
8	1-7	1-7	75
10	1-5	1-5	250
11	1-4	1-4	1090

Fig. 4. Demonstrates the relationship between *p* and the average number of hops *ahr* with varying *m* values.

Goal 4

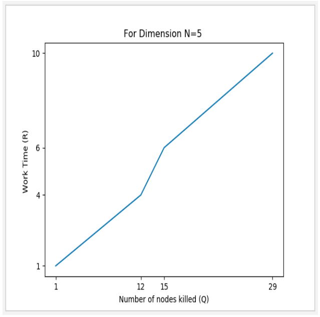
The introduction of an adversary affects the system metrics and subsequently affects the performance of the hypercube architecture. The main thread is given the capability of an adversary which takes the amount of time *r* to simulate its attack and begins to destroy *q* number of nodes in the system. It can be inferred from the experiments that for higher *r* values, the value of *q* is also higher.



q (nodes killed)	*r* (time worked by evil)
1	1
2	4
2	6
4	10

Fig. 5. Shows the relationship between *q* and *r* keeping *n* value constant

Figure 5 consists of a table having 4 set of values for q and r which are utilized in plotting the graph for a constant n value 2.



q (nodes killed)	*r* (time worked by evil)
1	1
12	4
15	6
29	10

Fig. 6. Shows the relationship between *q* and *r* keeping *n* value constant

Figure 5 consists of a table having 4 set of values for q and r which are utilized in plotting the graph for a constant *n* value 5.

V. Discussion

As can be seen from the experiments performed, variance in the number of dimensions *n*, number of files generated *m*, number of files retrieved *p* and the number of nodes attacked *q,* affects the system metrics which in turn affects the performance of the system.

As seen in the hops variance experiments, the efficiency of the system is higher when the average number of hops to store *ahs* is lower. As the number of nodes created *n* increases, the higher the number of files that can be generated *m* and subsequently higher number of hops required to generate the files thus lowering the system performance.

Similarly, when the number of files retrieved *p* is higher, the average number of hops to retrieve *ahr* also increases, thus lowering the efficiency of the system.

The disruption that occurs during an attack and death of the nodes in the system by the adversary might affect the performance of other nodes while they attempt to keep the system in a stable state. This lowers the efficiency immediately following an attack, until the individual nodes of the hypercube, once again reorganize in the presence of an adversary.

It can be inferred from the experimental results that there is a directly proportional relationship between *n* and *m*, *ahs* and *m, ahr* and *p, q* and *r.*

VI. Conclusion

In this paper, we study the dynamics of a peer-to-peer architecture based on the hypercube topology and provide the design of a file-sharing and lookup application. In this architecture, each node has an associated identifier that helps in the file transmission, storage and retrieval. The focus of this study was on analyzing the effects of varying metrics pertaining to the hypercube structure, file generation and node destruction on the system performance. The insight gained from quantifying the impact on the P2P systems after the introduction of an adversary and the results from the experiments, can be extremely useful in hypothesizing similar systems or extensions of this system. Built on top of a peer-to-peer overlay, the proposed design can further test the performance, reliability, fault tolerance and security by adding more variables and observing the causal effects between them.

References

[1] Rüdiger 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).

[2] Zulhasnine, Mohammed; et al. (2013). "P2P Streaming Over Cellular Networks: Issues, Challenges, and Opportunities". In Pathan; et al. Building Next-Generation Converged Networks: Theory and Practice. CRC Press. p. 99. ISBN 9781466507616.

[3] A Toce, A Mowshowitz:et al., "An efficient hypercube labeling schema for dynamic Peer-to-Peer networks", Journal of Parallel and Distributed Computing 102, 186-198

[4] Schlosser, M., Sintek, M., Decker, S., & Nejdl, W. (2002, July). HyperCuP - hypercubes, ontologies, and efficient search on peer-to-peer networks. In International Workshop on Agents and P2P Computing (pp. 112-124). Springer, Berlin, Heidelberg.

[5] Ghodsi, A. (2010). Multicast and Bulk lookup in structured overlay networks. In Handbook of Peer-to-Peer Networking (pp. 933-958). Springer, Boston, MA.

[6] Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. ACM computing surveys (CSUR), 36(4), 335-371.

[7] Chen, Y. W., & Lu, M. C. (2012). Design a Hypercube-Tree Structure for Peer-to-Peer Streaming. International Journal of Computer and Communication Engineering, 1(2), 81.

[8] Kuhn, F., Schmid, S., & Wattenhofer, R. (2005, February). A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In International Workshop on Peer-to-Peer Systems(pp. 13-23). Springer, Berlin, Heidelberg.