Replica Threshold Signature (RTS) Scheme for Reconfigurable Byzantine Fault Tolerant in P2P Systems

Mr.S.Annamalai M.E., M.B.A., (Ph.D)
Assistant Professor in CSE,
PGPCET, Namakkal,
Tamil Nadu, India.

Mrs.K.Hemalatha M.E (CSE)
Dept. of Computer Science & Engg,
PGPCET, Namakkal,
Tamil Nadu, India.

Abstract

Byzantine-fault-tolerant replication enhances availability and reliability of Internet services to store critical state and preserve even on attacks or software errors. Conventional Byzantine-fault-tolerant storage systems assumed static set of replicas, limitations in handling reconfigurations become issue in long-lived large-scale systems and membership likely to change during system lifetime. Existing presented set of services run automatically to tracks system membership periodically notifies other system nodes of membership changes Byzantine fault-tolerant and reconfigurable. It provides applications with a sequence of consistent views of system membership. It presented distributed hash table for provided atomic semantics even across changes in replica sets. The storage algorithms extend existing Byzantine quorum protocols to handle changes in the replica set. However, replica data was invalid and inconsistent cache across the sub system servants. The presented Replica Threshold Signature (RTS) scheme is used for Reconfigurable Byzantine Fault Tolerant in p2p Systems. It provides communication in terms of replica validity and cache consistency. It prevent smaller group from generating a signature and incorporation of RTS scheme allows for any member in the system to verify data object's content is valid for assumed that primary replicas are not faulty.

Keywords: Byzantine fault tolerance, membership service, dynamic system membership, and distributed hash tables.

I. INTRODUCTION

Byzantine fault tolerance refers to the capability of a system to tolerate Byzantine faults. It can be achieved by replicating the servant and by ensuring that all servant replicas reach an agreement on the total ordering of clients' requests despite the existence of Byzantine faulty replicas and clients. Such an agreement is often referred to as Byzantine agreement. Byzantine-fault-tolerant replication enhances the availability and reliability of Internet services that store critical state and preserve it despite attacks or software errors. However, existing Byzantine-fault-tolerant storage systems either assume a static set of replicas, or have limitations in how they handle reconfigurations. This can be problematic in long-lived, large-scale systems where system membership is likely to change during the system lifetime. In this paper, we present a complete solution for dynamically changing system membership in a large-scale Byzantine-fault-tolerant system. We present a service that tracks system membership and periodically notifies other system nodes of membership changes.

We demonstrate the utility of this membership service by using it in a novel distributed hash table (DHT) called dBQS [1] that provides atomic semantics even across changes in replica sets. dBQS (distributed Byzantine Quorum System) is interesting in its own right because its storage algorithms extend existing Byzantine quorum protocols to handle changes in the replica set, and because it differs from previous DHTs by providing Byzantine fault tolerance and offering strong semantics. We implemented the membership service and dBQS. This can be achieved by using an application level gateway, or a privacy firewall as described, to filter out illegal replies. A compromised replica may, however, replace a high entropy source to which it retrieves random numbers with a deterministic algorithm, and convey such an algorithm via out-of-band or covert channels to its colluding clients.

For replicas that use a pseudo-random number generator, they can be easily rendered deterministic by ensuring that they use the same seed value to initialize the generator. One might attempt to use the sequence number assigned to the request as the seed. Even though this approach is perhaps the most economical way to render replicas deterministic, it virtually takes the randomness away from the fault tolerant systems. In the presence of Byzantine clients, the vulnerability can be exploited to compromise the integrity of the system. Each system node n (either a client or a servant) maintains information about the current configuration, the last configuration (this is important for purposes of state transfer) and the current epoch number. We model these as the following per-node state variables.

Additionally, servants maintain their own view of the current value, a timestamp, and a signature (authenticating both the value and the timestamp) associated with the replicated variable, x: The timestamp space T is a totally ordered set used to determine the relative order of operations. We designed it such that different clients choose different timestamps, by appending the client id in the low-order bits. For now we will consider an infinite timestamp space, and later we will discuss some practical implications. There are other components of the state that we do not explicitly describe here, since they are only used for bookkeeping of the status of ongoing operations. In order to increase the threshold, new shares are generated from existing ones, by using the share-split procedure, and a fraction of cosigners The protocol that we describe here permits to increase the scheme's robustness from value t to value t + k, with k a positive odd number, any time it is executed.

II. LITERAUTURE SURVEY

Dynamo uses a synthesis of well known techniques to achieve scalability and availability: Data is partitioned and replicated using consistent hashing, and consistency is facilitated by object versioning. The consistency among replicas during updates is maintained by a quorum-like technique and a decentralized replica synchronization protocol [2]. A virtually synchronous execution is thus characterized by the following property: It will appear to any observant, any process using the system that all processes observed the same events in the same order. This applies not just to message delivery events, but also to failures, recoveries, group membership changes, and other events.

The essential issue in designing the toolkit is to ensure that the tools have orthogonal functionality, since it is this aspect that permits the programmer to break up an application into components that can be solved independently and extended gradually into a complete system [3]. Three features that distinguish Chord from many other peer-to peer lookup protocols are its simplicity, provable correctness, and provable performance. Chord is simple, routing a key through a sequence of other nodes toward the destination. A Chord node requires information about other nodes for efficient routing, but performance degrades gracefully when that information is out of date [4].

This protocol provides strong membership semantics, including total ordering of membership changes among all correct group members, provided that less than one-third of each group view is faulty. These faulty members are powerless to single-handedly alter the group members or prevent membership changes from occurring. This also ensures correct members control and consistently observe changes to group membership [5]. We distinguish three states of members: correct, stopped, and Byzantine. Correct members execute the protocols described in this paper faithfully. Stopped members are not executing any protocol steps. Byzantine members are not bound to the protocols. Members can switch between states at any time, which is commonly referred to as churn. Informally, Fireflies provides its correct members with a membership view that includes all members that have been correct for sufficiently long and excludes all members that have been stopped for sufficiently long [6].

Census is intended to be used in an asynchronous network like the Internet, in which messages may be corrupted, lost or reordered. We assume that messages sent repeatedly will eventually be delivered. We also assume nodes have loosely synchronized clock rates, such that they can approximately detect the receipt of messages at regular intervals. Loosely synchronized clock rates are easy to guarantee in practice, unlike loosely synchronized clocks [7]. The operations staff of all three services (Online, Content, and Read Mostly) use problem tracking databases to record information about component and service failures. Two of the services (Online and Content) gave us access to these databases, and one of the services (Read Mostly) gave us access to the problem post-mortem reports written after every major user visible service failure [8].

BFS (Breadth First Search) is a Byzantine-fault tolerant implementation of NFS (Network File Systems). BFS demonstrates that it is possible to use our algorithm to implement real services with performance close to that of an unreplicated service. This good performance is due to a number of important optimizations, including replacing public-key signatures by vectors of message authentication codes, reducing the size and number of messages, and the incremental checkpoint-management techniques [9].

COCA is the first system to integrate a Byzantine quorum system (used to achieve availability) with proactive recovery (used to defend against mobile adversaries which attack, compromise, and control one replica for a limited period of time before moving on to another). In addition to tackling problems associated with combining fault-tolerance and security, new proactive recovery protocols had to be developed [10]. The digital signature scheme in which the public key is fixed but the secret signing key is updated at regular intervals so as to provide a forward security property: compromise of the current secret key does not enable an adversary to forge signatures pertaining to the past. This can be useful to mitigate the damage caused by key exposure without requiring distribution of keys [11].

III. REPLICA THRESHOLD SIGNATURE (RTS) SCHEME FOR RECONFIGURABLE BYZANTINE FAULT TOLERANT IN P2P SYSTEMS

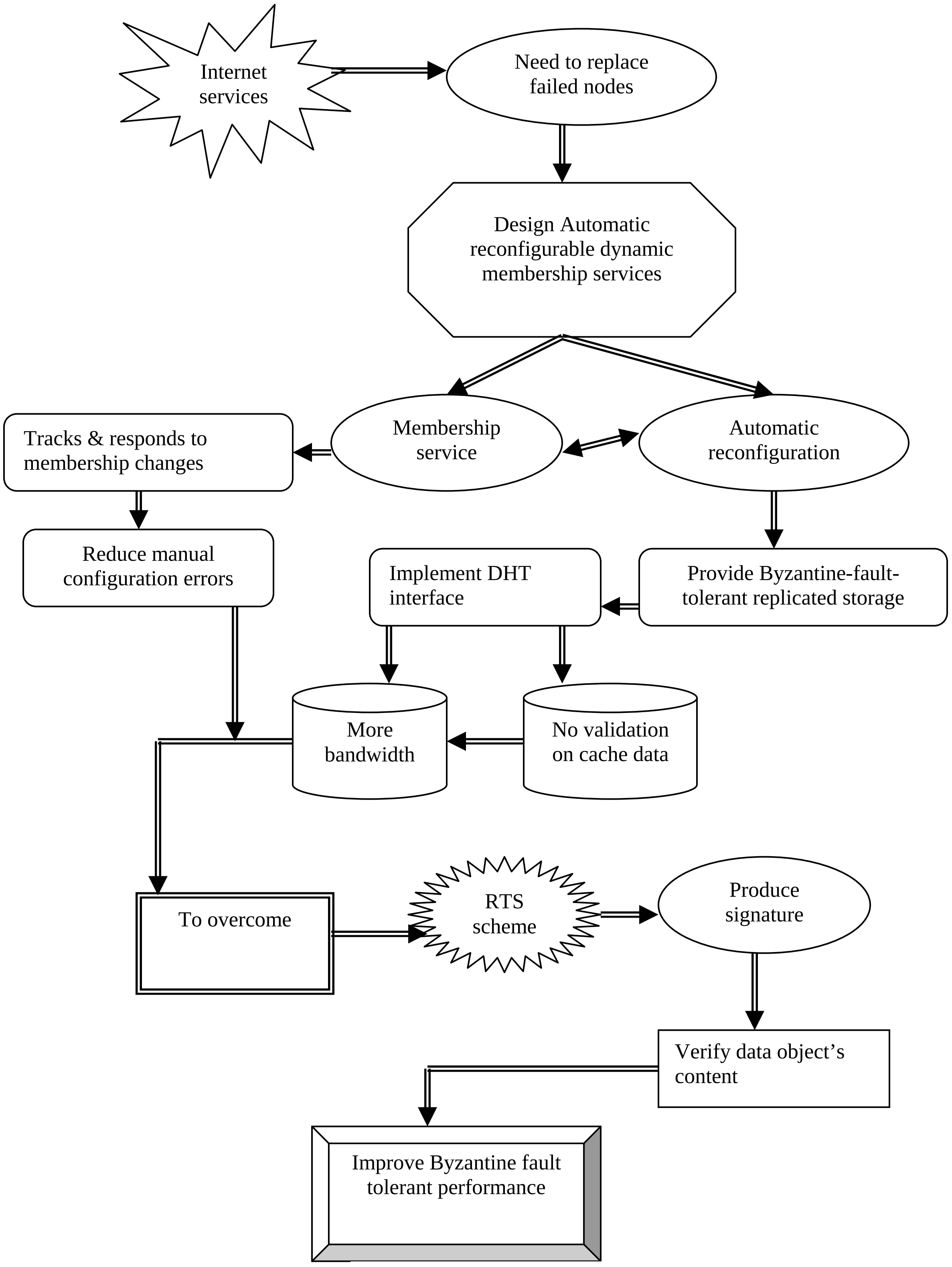
Internet services
Need to replace failed nodes
Design Automatic reconfigurable dynamic membership services
Membership service
Automatic reconfiguration
Tracks & responds to membership changes
Reduce manual configuration errors
Implement DHT interface
Provide Byzantine-fault-tolerant replicated storage
More bandwidth
No validation on cache data
To overcome
RTS scheme
Produce signature
Verify data object's content
Improve Byzantine fault tolerant performance

Figure 1. Architecture Diagram

The phases involved in the schemes are

- System and Application Services
- Membership Services
- Byzantine-fault-tolerant
- Dynamic replication and consistency
- Replica Threshold Signature

a) System and Application Services

System application services comprised of nodes that servants implementing storage service or clients consuming those services and it have two sets are disjoint. The nodes are connected by unreliable asynchronous network (Internet). The messages get lost, corrupted, delayed, duplicated or delivered out of order. It makes partial synchrony assumptions for live ness of services. Byzantine failure model is made faulty nodes behave arbitrarily. This compromised node remains compromised for entire life time of the system and services. The Byzantine faulty node expose secret information (private key), that cannot be recovered and continue to be trusted.

b) Membership Services

Membership service (MS) provides trusted source of membership information. It describes membership changes by producing configuration and identifies set of servants currently in the system. It exchange configuration information among nodes without possibility of forgery and authenticates using a signature. The MS produces configurations periodically not on after every membership change. The system moves in a succession of time intervals and it batch all configuration changes at the end of the epoch. Producing configuration periodically allows applications using MS to be optimized for long periods of stability to reduce associated propagation overhead. MS process includes membership change request, Probing, Ending epochs, and Freshness.

c) Byzantine-fault-tolerant

This system provides Byzantine fault tolerance for MS to implement with a group of replicas executing PBFT state machine replication. MS replicas run on servant nodes and size of the MS group is small and independent of system size. To implement ADD and REMOVE as PBFT operations. We consider the arguments of respective certificate, current set of members to be updated, and service state maintained by the MS. Freshness also implemented as PBFT operations and MS probes servants decides when the servants are faulty, when to end an epoch, propagates information about new configuration to all servants. PBFT prevents faulty members of MS from causing a malfunction at the same time ensure progress. Replicas probe independently proposes an eviction for a servant node missed probe responses. It sends eviction messages to other MS replicas that wait for signed statements from specified MS replicas agree to evict that node. The other MS replicas accept eviction.

d) Dynamic replication and consistency

Dynamic replication and consistency are storage applications handle reconfigurations using membership service. It design dBQS read/write block storage system based on Byzantine quorums. The dBQS uses input from MS to determine when to reconfigure and also it has strong consistent distributed hash table. DHT provide two types of objects, public key objects mutable and written by multiple clients, content-hash objects immutable once created. Data objects in dBQS have version numbers determine data freshness and identifiers are made to allow data to be self-verified. ID of public key object is a hash of the public key that used to verify integrity of the data. Each object is stored along with a signature that covers both data and version number.

When client fetches an object that integrity is checked by verifying signature using public key (validated by the ID of the object). Partition objects among system nodes using consistent hashing. Servants are assigned random IDs in the same ID space as objects. The replica group responsible for ID consistency that assignment of replica groups to objects. dBQS ensures concurrent accesses to entire set of public-key objects (atomic). All system operations appear to execute in a sequential order consistent with the real-time order in which operations actually execute.

e) Replica Threshold Signature

Replica Threshold Signature is made in a way that signs the results of subset replica of local cache collective and makes decisions for RTS to be optimal. Permit signature being produced till membership servants agree to cooperate. It is used to avoid signature generation by smaller groups and verify data object's replica content is valid. Members check the consistency of replica object content. The primary replicas are consistent across the servant throughout the lifetime of the services rendered. Member of the application services verify consistency of the replica object with number of inner ring members and time stamp for each decision. Number of consistent replica threshold is linear in size of the ring of subset replicas.

IV. EXPERIMENTAL RESULT AND DISCUSSIONS

Experiment is conducted on Reconfigurable Byzantine Fault Tolerant Storage Systems for evaluating the performance of Replica Threshold Signature (RTS) scheme is implemented in J2EE. Proposed Replica Threshold Signature (RTS) scheme is evaluated with different performance metrics and compares it with existing Automatic Reconfigurable Dynamic Membership Services. By comparing existing Automatic Reconfigurable Dynamic Membership Services, the experimental results show that Replica Threshold Signature (RTS) scheme not only have satisfactory performance. Also achieve stronger Responsive Replica Threshold Signature (RTS) scheme.

The performance of Replica Threshold Signature (RTS) scheme is evaluated by the following metrics.

- Cache Consistency Rate
- Reconfiguration Time
- Error Rate on Replica Storages

TABLE I: CACHE CONSISTENCY RATE

Storage Size (MB)	Existing Automatic Reconfigurable Dynamic Membership Services	Proposed Replica Threshold Signature (RTS)
10	130	180
20	112	170
30	98	140
40	85	125
50	60	105

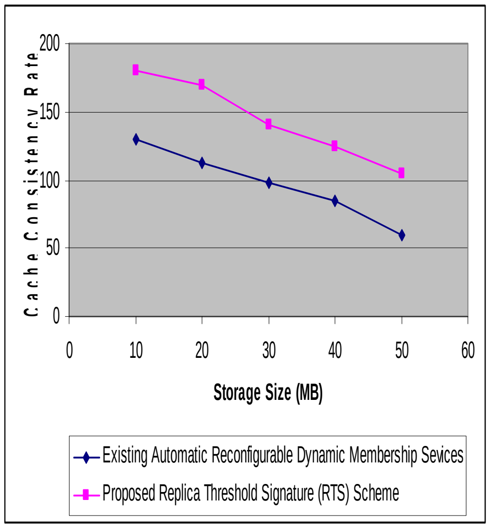

Figure 2. Cache Consistency Rate

Figure 2 demonstrates the cache consistency rate. X axis represents the storage size whereas Y axis denotes the cache consistency rate using both Automatic

Reconfigurable Dynamic Membership Services and our proposed Replica Threshold Signature (RTS) Scheme. When the size of storage decreased number of cache consistency rate also gets decreased. In the resistance rate in full delivery and a constant approximation factor to the optimum solution in the cache consistency rate. All the curves show a more of less yet steady descendant when storage size increases. Figure 2 shows better rate for cache consistency of Integrated Replica Threshold Signature (RTS) Scheme. Replica Threshold Signature (RTS) Scheme achieves 20% to 30% less cache consistency rate result.

TABLE II. VALIDATION STATE OF REPLICA

Membership Size (MB)	Existing Automatic Reconfigurable Dynamic Membership Services	Proposed Replica Threshold Signature (RTS)
20	2.4	1.3
40	2.9	1.9
60	3.7	2.2
80	3.7	2.8
100	4.1	3.1

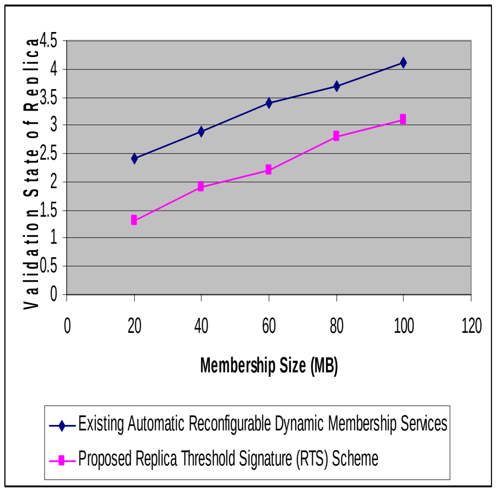

Figure 3. Validation State of Replica

Figure 3 demonstrates the validation state replica. X axis represents the membership size whereas Y axis denotes the validation state of replica using both the Automatic Reconfigurable Dynamic Membership Services and our proposed Replica Threshold Signature (RTS) Scheme. When the size of membership increased, validation state of replica also gets increases accordingly. The replica of validation state is illustrated using the existing Automatic Reconfigurable Dynamic Membership Services and our proposed Replica Threshold Signature (RTS) Scheme. Figure 3 shows better performance of Proposed Replica Threshold Signature (RTS) Scheme in terms of membership than existing Automatic Reconfigurable Dynamic Membership Services and Proposed Replica Threshold Signature (RTS) Scheme. Replica Threshold Signature (RTS) Scheme achieves 30to 50% less validation state of replica variation when compared with existing system.

TABLE III: ERROR RATE OF REPLICA STORAGE

Storage Size (MB)	Existing Automatic Reconfigurable Dynamic Membership Services	Proposed Replica Threshold Signature (RTS)
10	100	50
20	130	80
30	190	110
40	220	140
50	250	160

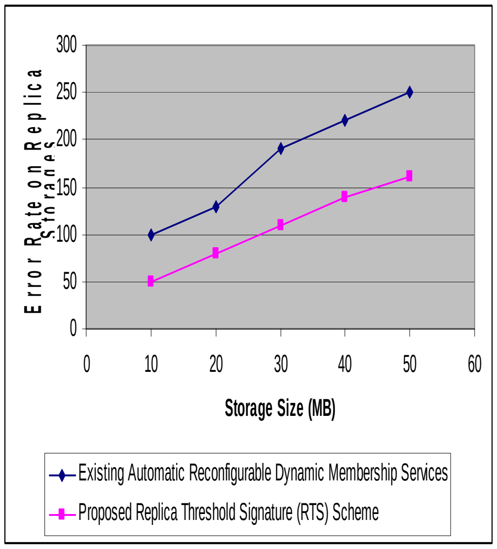

Fig 4: Error Rate of Replica Storages

Figure 4 demonstrates the error rate of replica storages. X axis represents storage size whereas Y axis denotes the error rate on replica storages using both the Automatic Reconfigurable Dynamic Membership Services and our proposed Replica Threshold Signature (RTS) Scheme. When the size of storage increased error rate on replica storages also gets increased. Figure 4 shows the effectiveness of error rate on replica storages over different size of storage than existing Automatic Reconfigurable Dynamic Membership Services and proposed Replica Threshold Signature (RTS) Scheme. Replica Threshold Signature (RTS) Scheme achieves 40% to 60% more error rate on replica storages when compared with existing schemes.

V. CONCLUSION

This paper presents a complete solution for building large scale, long-lived systems that must preserve critical state in spite of malicious attacks and Byzantine failures. We present a storage service with these characteristics called dBQS, and a membership service that is part of the overall system design, but can be reused by any Byzantine-fault tolerant large-scale system. The membership service tracks the current system membership in a way that is mostly automatic, to avoid human configuration errors. When membership changes happen, the replicated service has work to do: responsibility must shift to the new replica group, and state transfer must take place from old replicas to new ones, yet the system must still provide the same semantics as in a static system. We implement the Replica Threshold Signature (RTS) Scheme for perform a Byzantine fault tolerant in storage systems and collectively sign the results of subset replica of local cacher decisions.

REFERENCES

[1] Rodrigo Rodriguez, Barbara Liskov, Member, IEEE, Kathryn Chen, Moses Liskov, and David Schultz, "Automatic Reconfiguration for Large-Scale Reliable Storage Systems", IEEE transactions on Dependable and Secure Computing, VOL. 9, NO. 2, Mar. 2012.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W.Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," Proc. 21st ACM Symp. Operating Systems Principles, pp. 205-220, 2007.

[3] K. Birman and T. Joseph, "Exploiting Virtual Synchrony in Distributed Systems," Proc. 11th ACM Symp. Operating Systems Principles, pp. 123-138, Nov. 1987.

[4] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H.Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM, 2001.

[5] M. Reiter, "A Secure Group Membership Protocol," IEEE Trans. Software Eng., vol. 22, no. 1, pp. 31-42, Jan. 1996.

[6] H.D. Johansen, A. Allavena, and R. van Renesse, "Fireflies: Scalable Support for Intrusion-Tolerant Network Overlays," Proc.European Conf. Computer Systems (EuroSys '06), pp. 3-13, 2006.

[7] J. Cowling, D.R.K. Ports, B. Liskov, R.A. Popa, and A. Gaikwad, "Census: Location-Aware Membership Management for Large-Scale Distributed Systems," Proc. Ann. Technical Conf. (USENIX '09), June 2009.

[8] D. Oppenheimer, A. Ganapathi, and D.A. Patterson, "Why Do Internet Services Fail, and What Can Be Done About It?" Proc. Fourth USENIX Symp. Internet Technologies and Systems (USITS '03), Mar. 2003.

[9] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," Proc. Third Symp. Operating Systems Design and Implementation (OSDI '99), Feb. 1999.

[10] L. Zhou, F.B. Schneider, and R. van Renesse, "Coca: A Secure Distributed On-Line Certification Authority," ACM Trans. Computer Systems, vol. 20, no. 4, pp. 329-368, Nov. 2002.

[11] M. Bellare and S. Miner, "A Forward-Secure Digital Signature Scheme," Proc. 19th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '99), pp. 431-448, 1999.

S. Annamalai, completed M.E in Computer Science and Engineering. Now doing PhD under Peer to Peer Networking.

K. Hemalatha, doing M.E in Computer Science and Engineering.