Gossip Membership Management with Social Graphs for Byzantine Fault Tolerance in Clouds

Jongbeom Lim, Joon-Min Gil, Kwang-Sik Chung, Jihun Kang, Daewon Lee, Heonchang Yu

▶ **To cite this version:**

Jongbeom Lim, Joon-Min Gil, Kwang-Sik Chung, Jihun Kang, Daewon Lee, et al.. Gossip Membership Management with Social Graphs for Byzantine Fault Tolerance in Clouds. 11th IFIP International Conference on Network and Parallel Computing (NPC), Sep 2014, Ilan, Taiwan. pp.321-332, 10.1007/978-3-662-44917-2_27 . hal-01403099

HAL Id: hal-01403099
https://inria.hal.science/hal-01403099

Submitted on 25 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Gossip Membership Management with Social Graphs For Byzantine Fault Tolerance in Clouds*

JongBeom Lim[1], Joon-Min Gil[2], Kwang-Sik Chung[3], Jihun Kang[1], Daewon Lee[4], and Heonchang Yu[1]**

[1] Department of Computer Science Education, Korea University, Seoul, Korea
{jblim,k2j23h,yuhc}@korea.ac.kr
[2] School of Computer & Information Communications Engineering, Catholic University of Daegu, Daegu, Korea
jmgil@cu.ac.kr
[3] Department of Computer Science, Korea National Open University, Seoul, Korea
kchung0825@knou.ac.kr
[4] Department of General Education, SeoKyeong University, Seoul, Korea
daelee@skuniv.ac.kr

Abstract. As computer systems have become more complex and dynamic, unstructured and decentralized techniques serve as basic building blocks in large-scale systems such as cloud computing systems. In particular, we consider a gossip-based algorithm, one of the unstructured overlay construction techniques. In this paper, we propose a membership management mechanism using the gossip-based algorithm with social graphs for the Byzantine fault tolerance problem. Experimental results show that our membership management mechanism copes with Byzantine nodes effectively in a scalable way without a bottleneck in dynamic computing environments, requiring only $n \geq 2f + 1$ nodes.

1 Introduction

In recent years, the epidemic or gossip-based communication model has been employed in many applications in large-scale distributed systems and cloud computing systems. These applications include information dissemination [1], [2], clock synchronization [3], mutual exclusion [4], deadlock detection [5], termination detection [6], video streaming service [7], and BitTorrent (Tribler) [8]. In cloud computing environments, nodes can join or leave the system at will by virtue of virtualization technology [9]. Because of the characteristics of typical cloud computing environments, that is, the overlay network is often not fully connected and is constantly changing, the existing communication models are not able to suitably address reliability and scalability problems [10]. Therefore,

* This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea goverment (MEST) (No. NRF-2012R1A2A2A02046684).
** Corresponding author

as cloud computing matures, the gossip-based communication model has received significant attention because of its inherent ability to handle the dynamic behavior of nodes or resources [11].

Because modern gossip-based protocols use *local view*, which is a membership table that contains a small number of neighbor nodes, rather than maintaining full membership information of the system, the uniformity of peer sampling has become an important basic factor for evaluating the protocol. In this regard, several membership management mechanisms of the gossip-based protocols have been devised [12], [13], [14] to maintain the uniformity of peer sampling. Although biased peer sampling may not influence the correctness of the gossip protocol, it leads to performance degradation for applications. Furthermore, malicious or Byzantine nodes may subvert the system even though the number of Byzantine nodes is sufficiently small, and therefore, existing membership management mechanisms are not suitable for preserving the uniformity of random sampling.

In this paper, we propose a membership management solution over social graphs in the presence of Byzantine nodes. Although previous solutions focus on the uniform sampling of nodes, including Byzantine nodes, we endeavor to sample nodes within the set of correct nodes disregarding Byzantine nodes from local views. In brief, when a correct node encounters a suspicious (Byzantine) node, the correct node does not accepting the membership information of the suspicious node and leverages the pre-existing social graph for membership management.

The rest of this paper is organized as follows. We describe the gossip protocol, social graph, and Byzantine fault tolerance problem with related work in Section 2. In Section 3, we provide our system model and algorithms for membership management using social graphs. We present the results of performance evaluation in Section 4. Then, we conclude the paper in Section 5.

2 Related Work

An epidemic or gossip protocol is a method to communicate among uniquely identifiable nodes in a cycle-based fashion, inspired by the spread of disease. Diseases such as airborne diseases, contagious diseases, or HIV can be spread when individuals encounter others through networking connections. Another analogy of a gossip protocol can be found in the social behavior of persons. For example, if person P has just been updated for some data, P is willing to spread that information to other persons. Subsequently, P will contact some neighbors and try to push the data. In contrast, if person P has not yet obtained new data, P wants to be updated and P will try to obtain the data by pulling other neighbors. A gossip protocol guarantees message delivery with a high probability even if failures occur because of its inherent properties [15]. Refer to [16] and [17] for a correctness proof of the gossip protocol. The simplest form of the gossip protocol comes in two states: susceptible and infected. This form of the gossip protocol is called the SI model [18]. In the gossip protocol, each node maintains little

membership information, which is called *local view*, instead of full membership information in the system. Hence, the overlay network can be greatly simplified. At each cycle, a node selects o(fanout) gossip targets from its local view and then communicates with the gossip targets using one of the following methods: (1) push mode, (2) pull mode, and (3) push-pull mode.

As for membership management of the gossip protocol, several schemes have been proposed. In [12], the authors proposed the *view-shuffling* operation, where pairs of nodes regularly and continuously swap portions of their local views. Unfortunately, the naive version of the view-shuffling operation has some drawbacks in that the overlay network may be partitioned in some cases. Hence, an enhanced version of view shuffling has been proposed [13]. The difference between naive view shuffling and enhanced view shuffling is that in the enhanced version, when swapping local views, the initiator includes the *id* of the gossip target in the *sent view*, which will be included to the local view of the gossip target, and then replaces the *id* of the gossip target with its own *id* before transmission. This modification results in the uniformity of random sampling even when starting from a non-uniform distribution of nodes in the local views [19].

In Newscast [14], each node performs a view-swapping operation periodically, keeping only the most up-to-date local view entries of the union of the two local views. This idea is based on the assumption that nodes exhibit dynamic behavior, and therefore, the probability of existing in the local views of two nodes is high for newly joined nodes. In Brahms [20], the authors proposed a Byzantine-resilient and uniform peer sampling algorithm based on view shuffling, requiring multiple samplers and validators, where a unique hash function is used in each. Because of this, to obtain a uniform sample for a sampler, a sufficiently long sequence of shuffle operations is required. Moreover, the uniformity is valid only for one instance. In other words, another sequence of shuffling operations needs to be performed for another instance.

On the other hand, the membership management proposed in this paper does not require multiple samplers at each node, and the additional overhead for uniformity is minimized. Furthermore, the uniformity of local views is *always* evolving as the number of gossip cycles increases, retaining the previous uniformity regardless of different instances of membership management operations. In addition, as discussed, we endeavor to sample nodes within the set of correct nodes disregarding Byzantine nodes from local views, unlike the previous solutions, which focus on the uniform sampling of nodes including Byzantine nodes.

We consider social graphs to enhance the uniformity of peer sampling, disregarding Byzantine nodes. An informal definition of a social graph is a graph representation of an overlay network, of which every two nodes with a social relationship are connected through an overlay. In a typical gossip protocol, the overlay network is constructed only by local views of nodes. In addition to this, if Node A has relationships with Node G and Node H, Node A has an outdegree of seven (if the gossip protocol uses push mode).

Assume that Node A has data informative to its neighbors (i.e., nodes in a social view), but the data are sensitive and private. In such a case, Node A does not want to expose the data to nodes (although the data are beneficial to the nodes) other than its neighbors. Similarly, if Node A has sent sensitive and private data to its neighbor (Node G), Node A requires that Node G not send the data because the social relationships are not transitive. We note that the social relationships are symmetric. That is, if Node A trusts Node G, Node G also trusts Node A. We note that even though Node A and G have a trust relationship, they may not have contact information between them. For instance, if Node A contacts Node G, which does not have Node A in its own contact list, Node G will recognize Node A, and vice versa.

We use these properties of social graphs to solve the Byzantine fault tolerance problem. As far as confidentiality and privacy are concerned, explicit mechanisms should be employed when using social graphs. Because we focus on the Byzantine fault tolerance problem, these confidentiality and privacy mechanisms are beyond the scope of this paper. Several studies have been devoted to these mechanisms [21], [22].

3 Proposed Membership Management

In this study, we propose an enhanced version of gossip membership management with social graphs. The basic idea of our proposed solution is to utilize an existing social relationship in order to increase the expectations of the correct nodes in local views. More precisely, when a node encounters a suspicious node, the node utilizes its social neighbors to replace the suspicious node with a trustworthy node. As gossip cycles progress, a correct node may contact a Byzantine node, which outputs an incorrect decision value. In this case, we let the correct node perform a membership management algorithm that we provide. The pseudocode of the algorithm is provided in Section 3.2. In brief, in our membership management, the correct node contacts its social neighbor and then retrieves the social view of one of the social neighbors. Afterwards, the correct node replaces the Byzantine node information in the local view with the retrieved information.

One of the simplest forms that solve the Byzantine consensus problem is based on broadcast primitives. In this approach, the leader periodically sends a broadcast message to every node in the system, and then the leader waits until it receives all of the acknowledgements. Next, the leader performs the consensus algorithm to decide whether consensus is reached. If consensus is not reached (because some distributed computations are not finished), the leader sends another broadcast message. The drawbacks of this approach are: (1) the leader should remain stable during the whole epoch (single point of failure), (2) the message complexity of the algorithm is $O(n^2)$ (scalability problem), (3) it is not good at handling the dynamism of the system (not churn-resilient), and (4) no node (except for the leader) knows the system-wide information (not globally optimized).

To solve the global optimization problem, one can use gossip algorithms based on a broadcast primitive. In such a case, at each cycle, every node sends its local information to every other node in the system. Although the broadcast-based algorithms require fewer gossip cycles to reach consensus, the message complexity of the algorithms is $O(n^2)$ *at each cycle*. Unlike the previous approaches, our solution is not based on the broadcast primitive or the leader and is able to properly address the scalability problem in terms of the number of nodes even in the presence of Byzantine nodes. Furthermore, as gossip cycles progress, the system-wide information is distributed across the nodes in the system. To the best of our knowledge, several previous studies have dealt with the Byzantine fault tolerance problem in a dynamic system. The implicit assumptions of some previous studies are: no Byzantine nodes exist in the system, and the system is static (i.e., nodes are not allowed to join or leave).

3.1 System Model

There is a set of nodes or processes and all the nodes are functionally equivalent to each other. Henceforth, we use the terms "node" and "process" interchangeably. There is no notion of global memory. Therefore, message passing is the only way to communicate in the system. Communication channels are reliable but are not restricted to FIFO. In terms of failures, we assume that any node can be subject to Byzantine failures (i.e., they arbitrarily deviate from the specification of the algorithm intentionally or inadvertently, outputting an incorrect decision value by definition). In the worst case, a malicious Byzantine node performs covert activities in collusion with other Byzantine nodes to hinder or delay the objectives of other correct nodes. To prevent identification forgery, we use a digital signature scheme that uses public and private keys to sign and verify a message. That is, a node signs a message with a private key before transmission to a gossip target, and a receiver verifies a message using a public key of a sender. This guarantees the identity and the reputability of the signatory.

For Byzantine consensus, we consider the *interactive consistency* problem [23], where $n \geq 2f + 1$. In the problem in the presence of Byzantine nodes, each node sends and receives its `DecisionVector` by gossiping and checks `DecisionVector` in order to reach the consensus. If over half of the `DecisionVector` has non-empty elements, and their values are identical, the nodes can conclude the consensus value. We assume that Byzantine nodes exhibit malicious behavior. To be more precise, they send an empty `DecisionVector` except for their own elements. This behavior is the best effort of malicious Byzantine nodes, if we use a cryptographic scheme for `DecisionVector`.

3.2 Detailed Algorithms

To realize our proposed scheme, an additional data structure is required (i.e., social view). In our scenario, however, the additional overhead resulting from the data structure is marginal because the size of the social view is small compared to that of the local view. Recall that the size of the local view is significantly

Algorithm 1: Management of social view for P_i

```
1  begin at each cycle
2      if P_i makes a new social neighbor P_j then
3          if socialView_{P_i} is full then
4              if there is P_k that has less friendship than P_j then
5                  socialView_{P_i} ← socialView_{P_i} − P_k;
6                  socialView_{P_i} ← socialView_{P_i} ∪ P_j;
7          else
8              socialView_{P_i} ← socialView_{P_i} ∪ P_j;
9      if P_i breaks up with P_j then
10         socialView_{P_i} ← socialView_{P_i} − P_j;
```

less than that of the membership information of the system. We note that we consider the size of the social view a global system parameter with the same value for all nodes. In the following algorithms, a subscript indicates the owner of the data structure. We assume that P_i is a correct node in the algorithms.

Algorithm 1 shows the pseudocode for social view management. At each cycle, P_i manages its social view (`socialView`) based on relationships with other nodes. If P_i has a new social neighbor P_j, it tries to add P_j to `socialView` (line 2-8). During this phase, P_i checks the empty slot for P_j. If no empty slot is available in `socialView` (line 3), it tries to find P_k that has less friendship than that of P_j (line 4). If there is P_k that meets the condition, it replaces P_k with P_j (line 5-6). If an empty slot is available in `socialView`, it adds P_j to `socialView` (line 8). When P_i breaks up with P_j, it removes P_j from `socialView` (line 10).

We assume that no correct node deviates from the specification of the protocol. The gossip-based protocol consists of two threads: an active thread that initiates communication at each cycle and a passive thread that waits for incoming messages. The proposed membership management mechanism uses the Byzantine consensus algorithm running in the system. By inspecting the result of the decision value of the encountered node, individual nodes differentiate correct nodes and Byzantine nodes. When a correct node encounters a suspicious node, the correct node does not accept the membership information of the suspicious node and performs the membership management algorithm proposed in this paper. Furthermore, we assume that correct nodes do not violate trust relationships with others. In other words, no correct node will intentionally disclose membership information and create friendships with Byzantine nodes.

A full analysis of criteria determining friendships is out of the scope of this paper, because it depends on the specific characteristics of the applications using the gossip protocols. In fact, we can use an application-specific criterion that works best for the application. For example, Gossple [24] uses a *set item cosine similarity* metric to measure the friendships between nodes. If we apply the set item cosine similarity to measure the friendships, a node that has the larger metric value can replace the existing node. In addition, the application may

have a threshold value to determine the friendships for nodes. In this case, a node can remove (break up) one of the friends in `socialView` when a metric value between the two nodes is below the threshold value.

For the active thread of the gossip protocol, each time P_i selects a gossip target P_j it checks the decision value of P_j. We note that the `verify()` function returns *true* when two input parameters are identical; otherwise, it returns *false*. If `verify()` returns true, it assigns true to the `activate` variable. If the protocol is in push mode, P_i includes its own *id* in the `sendingView` and then sends this view to P_j. If the protocol is in pull mode, it tries to receive the `sendingView` from P_j and then updates its local view with the received view.

When the `verify` function returns false, P_i neither sends its view information to P_j nor receives the view information from P_j because `sendingView` from P_j may contain harmful information that pollutes the local view of P_i with Byzantine nodes. At this stage, P_i sets the `activate` variable to false and then selects one of the social neighbors from its `socialView`. After selecting the social neighbor, P_i tries to receive the `socialView` of the social neighbor. Then, P_i selects P_k, that is, one of the nodes from the `socialView` of the social neighbor. Afterwards, P_i replaces P_j with P_k in its `localView`. If P_i has no social neighbor, or `rand` function returns null, P_k cannot be inserted into `localView`. For brevity, this checking procedure is omitted. Lastly, P_i performs the `ByzantineConsensusAlgorithm()` function or not, based on the `activate` value.

For the passive thread of the gossip protocol, whenever P_i is selected from another node P_j it first checks `requestType`. If `requestType` is for a local view, P_i compares the `decisionValue` of P_j with its own value. If the two values are coherent, it accepts the `sendingView` of P_j and updates its local view with the received view in push mode. If the protocol is in pull mode, P_i includes its own *id* in `sendingView` and sends the view to P_j. If `requestType` is for the social view, and P_j is a friend, P_i sends its `socialView` to P_j.

4 Evaluation

In this section, we provide performance results of our membership management mechanism using social graphs. We do not include Shuffling [12] and Newscast [14] because those methods cannot tolerate Byzantine nodes even if the number of the Byzantine nodes is negligible. In our observation, Shuffling places less than 30% of the correct nodes in decision vectors on average when performing the Byzantine consensus algorithm if the Byzantine probability is 0.1, and the shuffle ratio is 50%. Newscast can be considered as view shuffling with a shuffle ratio of 100%. In other words, Newscast is more vulnerable to Byzantine nodes compared to Shuffling.

4.1 Experimental Settings

Table 1 shows the parameters and their values used in the evaluations. We note that the numbers in parentheses are the default values unless specified otherwise.

Table 1: Evaluation parameters and their values (numbers in parentheses are the default values unless specified otherwise)

Parameter	Value
Number of nodes	10^4
Gossip mode	Push-pull
Size of local view	20
Size of social view	8
Fanout	1
Gossip cycles per instance	20
Byzantine probability	0.1, 0.2, 0.3, (0.4)

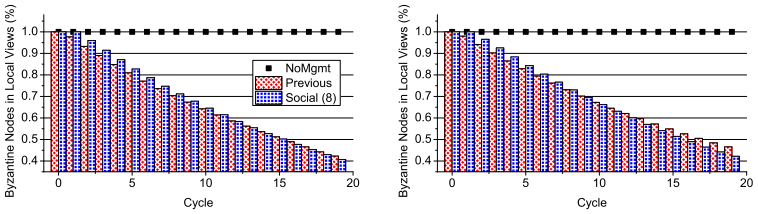
Because we need at least $f + 1$ correct nodes, the Byzantine probability is not configured to be higher than 0.5. Note also that because the probability is a measure of the expectation that an event will occur, the actual number of Byzantine nodes will be different from the number calculated with the Byzantine probability parameter.

Starting with the initial overlay network and local views in the presence of Byzantine nodes, we show how our membership management based on social graphs improves the uniformity of peer sampling. Then, we detail the effects on the local view to show how our proposed solution can improve the occurrence of correct nodes in local views. We note that we only show the results for correct nodes in the system because the results for Byzantine nodes are meaningless. Lastly, to show the scalability of the proposed approach, we present performance results by increasing the number of nodes exponentially. There were three objects for comparison: the default Byzantine consensus algorithm without membership management (NoMgmt), membership management performing random node sampling when a node encounters Byzantine nodes (Previous); and our membership management with social graphs when the social view size is 8 (Social(8)).

4.2 Performance Results

One of the design goals of our membership management is to reduce the possibility that a correct node contacts Byzantine nodes. Thus, we measured the number of Byzantine nodes in the local view to see how effectively our proposed membership management copes with Byzantine nodes. We assume that local views of individual nodes contain gossip partners selected at random from the system. Therefore, Byzantine nodes are in the local views of the correct nodes. Figures 1, 2, and 3 show the normalized percentages of Byzantine nodes in local views.

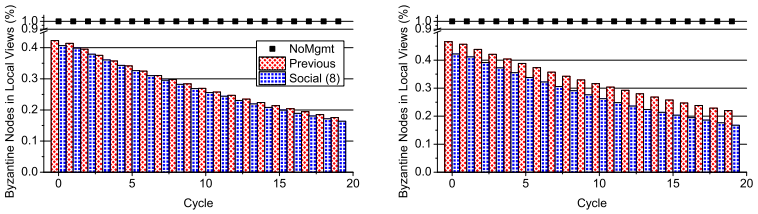
Figure 1 shows the results of the first instance. Because NoMgmt has no facility to perform membership management, the number of Byzantine nodes in local views is the same during the whole execution of the Byzantine consensus algorithm. Compared with Previous and Social(8), the percentage of Byzantine



(a) Byzantine probability: 0.1 (b) Byzantine probability: 0.2

(c) Byzantine probability: 0.3 (d) Byzantine probability: 0.4

Fig. 1: Normalized percentage of Byzantine nodes in local views (first instance)

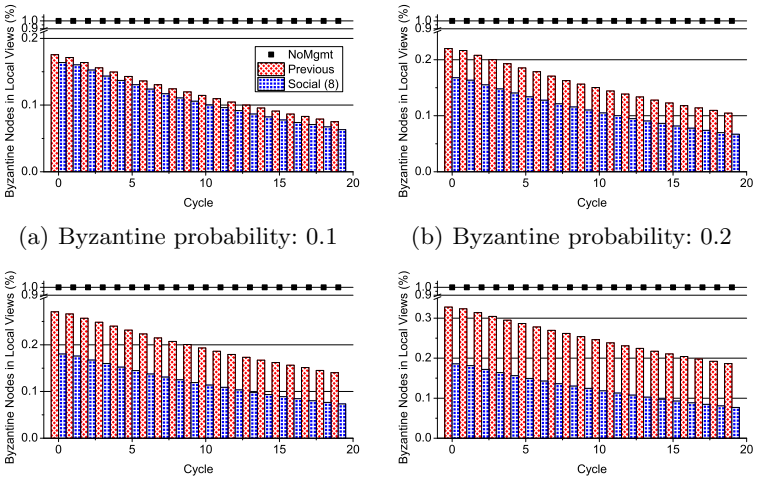


(a) Byzantine probability: 0.1 (b) Byzantine probability: 0.2

(c) Byzantine probability: 0.3 (d) Byzantine probability: 0.4

Fig. 2: Normalized percentage of Byzantine nodes in local views (second instance)

nodes in local views decreases as the gossip cycle proceeds in both methods, whereas Social(8) results in a greater reduction in the number of Byzantine nodes in local views than Previous in late cycles. It is interesting to note that in early cycles, Social(8) has a higher percentage of Byzantine nodes in local views than Previous. The reason for this phenomenon is that some nodes have no social neighbors. In other words, when P_i tries to select `socialNeighbor`, this cannot



(a) Byzantine probability: 0.1 (b) Byzantine probability: 0.2

(c) Byzantine probability: 0.3 (d) Byzantine probability: 0.4

Fig. 3: Normalized percentage of Byzantine nodes in local views (third instance)

be accomplished because the node has no social neighbors. Similarly, if P_i, who has some social neighbors, tries to receive `socialView` from `socialNeighbor`, it is possible that `socialNeighbor` has no social neighbors except P_i. In this case, no local view exchange is executed.

Figure 2 shows the results of the second instance. As in Figure 1, NoMgmt shows no change in the Byzantine nodes in local views. As the gossip cycle proceeds, and the Byzantine probability increases, Social(8) has a lower percentage than Previous and always outperforms Previous. By the specification of the algorithms, the nodes have a greater chance to exchange their local views when they encounter Byzantine nodes frequently. In this regard, there is a trade-off between the number of Byzantine nodes and the probability of exchanging local views. When the number of Byzantine nodes is small, the probability of performing membership management is low. Conversely, if the number of Byzantine nodes is large, the probability of performing membership management is high, and there are a number of local view slots to be changed.

Figure 3 shows the results of the second instance. As in Figures 1 and 2, Social(8) outnumbers Previous in the total reduction of Byzantine nodes, and the performance gap is greater with a larger Byzantine probability. At cycle 20, the percentages of Byzantine nodes in local views for Previous (resp. Social(8)) are approximately 7.49% (resp. 6.31%), 10.47% (resp. 6.69%), 14.02% (resp. 7.32%), and 18.64% (resp. 7.69%) when the Byzantine probability is 0.1, 0.2, 0.3, and 0.4, respectively. This means that our proposed membership management has 1.19, 1.56, 1.92, and 2.42 times fewer Byzantine nodes in local views then Previous when the Byzantine probability is 0.1, 0.2, 0.3, and 0.4, respectively.

5 Conclusion

In this paper, we have presented a membership management mechanism based on social relationships on the gossip overlay for the Byzantine fault tolerance problem. Rather than utilizing a traditional control method, where the centralized medium monitors the system and performs corrective functions, we let each node perform membership management with social graphs in a self-organizing way. Our self-organized construction of membership management using social graphs provides scalability, reliability, and resiliency in the presence of Byzantine nodes. The experimental results show that our membership management mechanism for Byzantine fault tolerance is globally optimized as the gossip cycle proceeds. Furthermore, our proposed membership management surpasses existing methods and effectively eliminates Byzantine nodes in the view of other nodes.

References

1. Lim, J., Lee, J., Chin, S., Yu, H.: Group-based gossip multicast protocol for efficient and fault tolerant message dissemination in clouds. In Riekki, J., Ylianttila, M., Guo, M., eds.: Advances in Grid and Pervasive Computing. Volume 6646 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 13–22
2. Antulov-Fantulin, N., Lani, A., tefani, H., iki, M.: Fastsir algorithm: A fast algorithm for the simulation of the epidemic spread in large networks by using the susceptibleinfectedrecovered compartment model. Information Sciences **239**(0) (2013) 226 – 240
3. Iwanicki, K., van Steen, M., Voulgaris, S.: Gossip-based clock synchronization for large decentralized systems. In: Proceedings of the Second IEEE international conference on Self-Managed Networks, Systems, and Services. SelfMan'06, Berlin, Heidelberg, Springer-Verlag (2006) 28–42
4. Lim, J., Chung, K.S., Chin, S.H., Yu, H.C.: A gossip-based mutual exclusion algorithm for cloud environments. In Li, R., Cao, J., Bourgeois, J., eds.: Advances in Grid and Pervasive Computing. Volume 7296 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 31–45
5. Lim, J., Suh, T., Yu, H.: A deadlock detection algorithm using gossip in cloud computing environments. In Han, Y.H., Park, D.S., Jia, W., Yeo, S.S., eds.: Ubiquitous Information Technologies and Applications. Volume 214 of Lecture Notes in Electrical Engineering. Springer Netherlands (2013) 781–789
6. Lim, J., Chung, K.S., Gil, J.M., Suh, T., Yu, H.: An unstructured termination detection algorithm using gossip in cloud computing environments. In Kubtov, H., Hochberger, C., Dank, M., Sick, B., eds.: Architecture of Computing Systems ARCS 2013. Volume 7767 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 1–12
7. Chu, Y.h., Ganjam, A., Ng, T.S.E., Rao, S.G., Sripanidkulchai, K., Zhan, J., Zhang, H.: Early experience with an internet broadcast system based on overlay multicast. In: Proceedings of the annual conference on USENIX Annual Technical Conference. ATEC '04, Berkeley, CA, USA, USENIX Association (2004) 12–12
8. Zeilemaker, N., Capotă, M., Bakker, A., Pouwelse, J.: Tribler: P2p media search and sharing. In: Proceedings of the 19th ACM international conference on Multimedia. MM '11, New York, NY, USA, ACM (2011) 739–742

9. Mahajan, K., Makroo, A., Dahiya, D.: Round robin with server affinity: A vm load balancing algorithm for cloud based infrastructure. Journal of Information Processing Systems **9**(3) (2013) 379–394
10. Matos, M., Sousa, A., Pereira, J., Oliveira, R., Deliot, E., Murray, P.: Clon: Overlay networks and gossip protocols for cloud environments. In: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I. OTM '09, Berlin, Heidelberg, Springer-Verlag (2009) 549–566
11. Wuhib, F., Stadler, R., Spreitzer, M.: A gossip protocol for dynamic resource management in large cloud environments. Network and Service Management, IEEE Transactions on **9**(2) (2012) 213–225
12. Stavrou, A., Rubenstein, D., Sahu, S.: A lightweight, robust p2p system to handle flash crowds. Selected Areas in Communications, IEEE Journal on **22**(1) (2004) 6–17
13. Voulgaris, S., Gavidia, D., Steen, M.: Cyclon: Inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management **13**(2) (2005) 197–217
14. Tölgyesi, N., Jelasity, M.: Adaptive peer sampling with newscast. In: Proceedings of the 15th International Euro-Par Conference on Parallel Processing. Euro-Par '09, Berlin, Heidelberg, Springer-Verlag (2009) 523–534
15. Ganesh, A., Kermarrec, A.M., Massoulie, L.: Peer-to-peer membership management for gossip-based protocols. Computers, IEEE Transactions on **52**(2) (feb. 2003) 139 – 149
16. Allavena, A., Demers, A., Hopcroft, J.E.: Correctness of a gossip based membership protocol. In: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing. PODC '05, New York, NY, USA, ACM (2005) 292–301
17. Gurevich, M., Keidar, I.: Correctness of gossip-based membership under message loss. In: Proceedings of the 28th ACM symposium on Principles of distributed computing. PODC '09, New York, NY, USA, ACM (2009) 151–160
18. Newman, M.: Networks: An Introduction. Oxford University Press, Inc., New York, NY, USA (2010)
19. Busnel, Y., Beraldi, R., Baldoni, R.: On the uniformity of peer sampling based on view shuffling. Journal of Parallel and Distributed Computing **71**(8) (2011) 1165 – 1176
20. Bortnikov, E., Gurevich, M., Keidar, I., Kliot, G., Shraer, A.: Brahms: Byzantine resilient random membership sampling. Comput. Netw. **53**(13) (August 2009) 2340–2359
21. Schiavoni, V., Riviere, E., Felber, P.: Whisper: Middleware for confidential communication in large-scale networks. In: Distributed Computing Systems (ICDCS), 2011 31st International Conference on. (2011) 456–466
22. Singh, A., Urdaneta, G., van Steen, M., Vitenberg, R.: Robust overlays for privacy-preserving data dissemination over a social graph. In: Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on. (2012) 234–244
23. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2) (April 1980) 228–234
24. Bertier, M., Frey, D., Guerraoui, R., Kermarrec, A.M., Leroy, V.: The gossple anonymous social network. In Gupta, I., Mascolo, C., eds.: Middleware 2010. Volume 6452 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 191–211