Impact of Node Cheating on Gossip-Based Protocol

Nan Zhang[1], Yuanchun Shi[1], Bin Chang[1]

[1] Dept. of Computer Science and Technology, Tsinghua University,
Beijing, China
{z-n04@mails., shiyc@, cb99@mails.}tsinghua.edu.cn

Abstract. Gossip-based protocol has been widely adopted by many large-scale multicast applications. In this paper, we study the impact of node cheating on decentralized gossip-based protocol. We mainly focus on two cheating strategies, one is to increase the subscription request sending times, and the other is to increase the *PartialView* size. We establish a cheating model of nodes for gossip-based protocol, and evaluate the system performance when node cheating happens. In the simulations, we analyze the impact of node cheating on a representative gossip-based protocol, SCAMP (Scalable Membership Protocol, a decentralized, gossip-based protocol). The results show that node cheating makes considerably negative effects on the system performance, and there exists a delicate relationship between the percentage of cheating nodes in the system and the benefit they can gain. The study results also show that cheating behaviors should be paid much more attention during the gossip-based protocol design in future.

Keywords: ALM, gossip-based protocol, node cheating, reliable.

1 Introduction

The innovation and progress of Internet-wide distributed applications is driving the need for scalable multicast mechanism which can provide a reliable group communication [1]. There exist two kinds of multicast technologies: one is IP multicast [2], which achieves in the network-level, and is not currently widely deployed. The other is Application Level Multicast (ALM), which achieves in the application-level, without any support from the network. So ALM has been proposed mainly as a way to alleviate to the lack of deployment of IP multicast, and has been an active research issue at present.

Unlike IP multicast, ALM replicates and forwards data in application level, and uses unicast to a group of receivers. So there is a trade-off between the efficiency of IP multicast and the ease of deployment of group communication [3]. Compared with IP multicast, ALM is more scalable and can provide more semantics. Therefore, it is feasible to design suitable models and management protocols to achieve large-scale reliable multicast.

Existing ALM management protocol can be divided into two categories: one is tree-based management protocol, which constructs a multicast delivery tree among end hosts [4], [5], [6]. This kind of topology is easy to manage, and the control

overhead is also quite low. However, it is not robust under the situation that group membership changes frequently. The other kind of management protocol is gossip-based, where each member is in charge of forwarding each message to a set of other, randomly chosen, group members. And each member can receive messages from more than one node [7], [8], [9]. This mechanism using of redundant messages provides the reliability in face of node crashes and high packet loss rates in the network. It has been proven that the load on each node increases only logarithmically with the size of the group, so gossip-based ALM protocol becomes an attractive alternative to achieve a scalable and reliable ALM.

In gossip-based ALM protocol, new nodes join the group by sending a subscription request to an arbitrary member. Each participant member in the protocol maintains two lists: one list of nodes it sends gossip message to and the other list of nodes that it receives gossip message from. Obviously, less the size of the former list is, less cost it will pay for maintaining the information as well as replicating and forwarding data; and larger the size of latter list is, more message providers the node will have, which means it will receive messages with less delay, and obtain a higher quality of service.

The important point here is that there is an opportunity for nodes to try and improve their performance in the gossip-based ALM system by sending more subscription requests during its joining process, and rejecting more subscription requests from others as many as possible. So as to increase the size of list of nodes that it receives gossip message from, diminish the size of list of nodes it sends gossip message to, and minimize its replication burden. These cheating behaviors may cause the structure change and thus the low performance of the management protocol. It also affects the normal propagation of multicast data, leads to more control overhead of computation, storage and communication.

In the rest of this paper, we will study the effects of cheating strategies on the gossip-based ALM protocol. In Section 2, some related works are reviewed. In Section3, the cheating model we defined will be introduced in detail. Section 4 presents our simulation study, while Section 5 concludes with a summary of our observation and some recommendations.

2 Related works

The impact of node cheating on tree-based ALM protocols has been greatly discussed recently. This section will give a brief overview of some analyses which have been done.

Authors in [3] study the impact of cheating nodes in four representative tree-based ALM protocols: HBM (a protocol based on a centralized approach), TBCP (a distributed, tree first protocol), NICE (a distributed, tree first protocol based on clustering) and NARADA (a mesh first protocol). They focus on selfish nodes acting independently, cheating about their distance measurements during the constructing the tree. They choose the *stress_ratio* and *stretch_ratio* as indicators to characterize the intrinsic performance of the ALM protocols. The simulation results show that simple cheating strategies always have negative impact, either on the performance of

the tree as perceived by its nodes (both cheats and honest nodes), or on the underlying physical network, or both.

Receiver cheating also brings considerably negative effects on the stability of ALM tree [12]. Dan Li established a cheating model and discussed the relationship between receiver cheating and the ALM tree's stability in detail. According to the simulation results, receiver cheating will not only cause much additional overhead for ALM to reconstruct the multicast tree, but also add more burdens to honest receivers, especially the source node.

Node cheating has become an important issue in the ALM researches, both tree-based ALM protocols and gossip-based protocols. However, as far as we know, compared with many discusses on the impact of node cheating on the tree-based ALM protocol, there is no research on the impact of node cheating on the gossip-based one. In this paper, this topic will be discussed in detail.

3 Cheating Model for Gossip-based ALM Protocol

In this section, we first introduce the mechanisms of decentralized gossip-based ALM protocol briefly, and then present the cheating model we defined for analyzing the impact of node cheating on gossip-based ALM protocol in detail.

3.1 Decentralized Gossip-based ALM Protocol

Gossip-based protocol uses randomization to reliably broadcast messages in the group. It has been used in various applications such as live media streaming [10], publish-subscribe systems [13] and so on. It provides probabilistic guarantees of delivery which degrade gracefully in the presence of loss links or fail nodes. By the difference of control manner, there are centralized gossip-based protocol and decentralized gossip-based protocol two types. The centralized means that each node should have global knowledge of membership, which greatly affects its scalability. While decentralized gossip-based ALM protocol only has to provide each node with a partial random view of the system. It requires less on memory and synchronization, which is much more scalable than centralized one, and is adopted by most large-scale multicast applications. In this paper, we analyze the impact of node cheating strategies on a simple and fully decentralized gossip-based protocol, SCAMP [8], which is also a representative one.

In SCAMP, a node maintains two lists:

- *PartialView* – records nodes it sends gossip messages to.
- *InView* – records nodes it receives gossip messages from.

New nodes subscribe (join) the group as follows: New nodes send a subscription request to an arbitrary member, called a *contact*, they start with a *PartialView* consisting of just their contact. When a node receives a new subscription request, it forwards the new node-id to all members of its own *PartialView*. It also creates *c* additional copies of the new subscription and forwards them to randomly chosen nodes in its *PartialView*. When a node receives a forwarded subscription, provided

the subscription is not already present in its list, it accepts the new subscriber in its *PartialView* with a probability *P* which depends on the size of its view ($P = 1/(1 + size\ of\ PartialView)$). If it doesn't keep the new subscriber, it forwards the subscription to a node randomly chosen from its *PartialView*. These forwarded subscriptions may be kept by the neighbors or forwarded, but are not destroyed until some nodes keep them.

The earlier work [11] shows that the probability that a notification reaches everyone exhibits a sharp threshold at $\log(n)$. And SCAMP is proven to has the following desirable properties: If new nodes join by sending a subscription request to a member chosen uniformly at random from existing members, then the system configures itself toward *PartialView* of *size* $(c+1)\log(n)$ on average (*n* is the number of nodes in the system). Therefore, SCAMP can guarantee the message to be reliably propagated to all group members.

Messages are propagated as follows: When a node generates a massage, it sends it to a random subset of nodes in its *PartialView*. When any node receives a message for the first time, it does the same. Obviously, larger size of *InView* and smaller size of *PartialView* is the thing every node desires, because it enables the node not only to obtain a high quality of service with low delay, but also not to provide much to others.

For the sake of gaining more benefits, nodes may make cheating in the joining process, and we refer this kind of nodes as "cheats". In this paper, the cheating strategies adopted are as follows: When sending subscription requests, cheats will subscribe more than once, so as to largen the *InView* size and to get a reliable service. Similarly, on receiving a subscription request, cheats make cheating in the *PartialView* size, so as to reduce the probability of becoming the provider of other nodes and have less replicating and forwarding burden. This paper will discuss the impact of these two cheating strategies in detail. For ease of exposition, we establish a cheating model to quantify it.

3.2 Cheating Model

Here we make some definitions as follows:

Definition 1. Subscription Cheating Degree. When a new node sends subscription request to join the group, the increased time to the actual time (once), is defined as Subscription Cheating Degree, which is denoted by *s*.

Definition 2. PartialView Cheating Degree. When receiving a subscription request, the node calculates the acceptance probability by the size of its *PartialView* ($P = 1/(1 + size\ of\ PartialView)$), the increased value of its *PartialView* size to the actual value is defined as PartialView Cheating Degree, which is denoted by *p*.

Definition 3. System Cheating Degree. In the gossip-based ALM system, the percentage of cheats out of all nodes is defined as System Cheating Degree, which is denoted by *t*.

Obviously, $s \geq 1$, $p \geq 1$, and $0 \leq t \leq 100\%$, different *t* with different *s* and *p* will lead to different impacts on the protocol. We denote this gossip-based ALM protocol system with cheats as $G(n, s, p, t)$, where *n* is the number of nodes in the system. To have a better view of the impact of node cheating on gossip-based ALM protocol, we divide nodes into a group of cheats and a group of honest nodes, and then analyze each group separate.

Cheat group will destroy the balance of the whole system, in order to seek their benefit. In the gossip-based ALM protocol system, the *InView* size of nodes can be considered as a measurement of the service quality it gains, and the *PartialView* size can be considered as a measurement of the service it should provide. Therefore, the "Benefit" cheat group gains can be defined as follows:

Definition 4. Cheat Group Benefit Degree. In a $G(n, s, p, t)$, we define $\sum_i (sizeof\ InView_i - sizeof\ PartialView_i)$ as Cheat Group Benefit Degree, denoted by $\theta(n, s, p, t)$, where *i* is the number of cheats in the group, *i=tn*.

Based on definition 4, $\theta(n, s, p, t) < 0$ shows that cheat group does not gain any benefit from the cheating behaviors, but a victim instead. Less $\theta(n, s, p, t)$ is, more damage the cheat group suffers. On the contrary, $\theta(n, s, p, t) > 0$ means that the cheat group gets benefit from the cheating behaviors. More $\theta(n, s, p, t)$ is, more it benefits, then more damage the honest group suffers.

4 Simulations

We have tested 25 groups of 50,000 nodes each in a gossip-based session. The results are the average values of all the 25 groups. Each group was tied with a subscription cheating degree *s* of 5, 10, 15 and 20 respectively, a PartialView cheating degree *p* of 10, 20 and 50 respectively, and a system cheating degree *t* of 0%, 20%, 40%, 60%, 80% and 100% respectively. To study the impact of node cheating on gossip-based ALM protocol, we show the distributions of *PartialView* size, and *InView* size of a 50,000 node gossip-based ALM protocol SCAMP system with no cheats first.

From Fig. 1, we find that the SCAMP system with no cheats can achieve an average *InView* and *PartialView* size of $\log(n)$, where n is the size of the group ($\log(50000) = 10.8$). It is similar with the result in [9]. The earlier work [11] has shown that a *PartialView* size of this order can ensure that gossip is successful with high probability. Then what will this gossip-based ALM system be when node cheating happens?

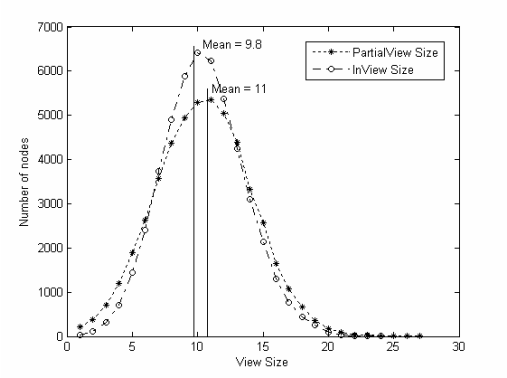


Fig. 1. Distribution of the PartialView size and InView size of a 50,000 node SCAMP system.

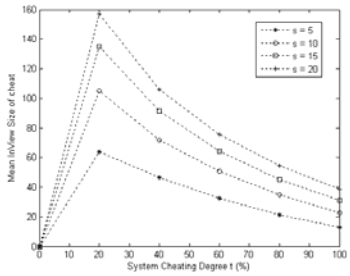
We analysis the *InView* size, *PartialView* size of cheat group and honest group respectively, as well as cheat group benefit degree $\theta(n, s, p, t)$, against different node cheating parameters.

4.1 Subscription Cheating Degree Impact

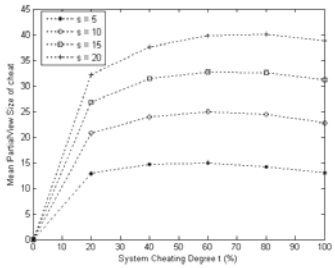
The value of cheat *InView* size, *PartialView* Size and $\theta(n, s, p, t)$ of the gossip-based ALM protocol against *s* and *t* when $p = 20$ are illustrated in Fig 2 (a) to (c).

As shown in Fig. 2 (a), with a fixed *t*, cheat *InView* Size increases as *s* grows. It is because that cheat sends more subscription requests, more nodes will accept the requests and become a member of its *InView*. However, we find that with a fixed *s*, the mean *InView* Size of cheat increases when $0\% \le t \le 20\%$, and when $t > 20\%$, the mean *InView* size of cheat goes down. This can be explained as follows: When the System Cheating Degree *t* is small, a majority of nodes are honest nodes, who integrate the new subscriber in their *PartialView* with an honest probability, *P*. So bigger *t* is, more honest nodes will be caught with chaff, and the cheat chance will go up rapidly. However, when $t > 20\%$, most nodes are cheats, the probability of cheat behavior between cheats will go up, which will counteract the cheating effects to some extent. Many subscription requests cannot be accepted by any nodes, and finally be discarded (To avoid a subscription is forwarded an infinite number of times, we limit that when a node has received the same request more than 10 times, it simply discards the thread, same as [8]). So the mean *InView* size of cheat decreases.

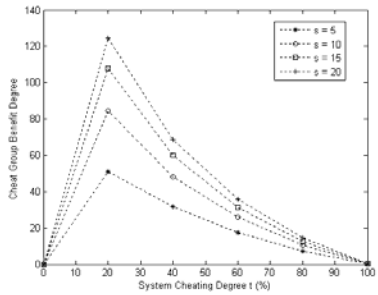
From Fig. 2 (b), we find that when *t* is settled, the mean of cheat group *PartialView* size grows as *s* increases, which is similar to Fig. 1 (a). But as the System Cheating Degree *t* is higher, the mean of *PartialView* size does not have much change. The reason is that the factor that influences the node *PartialView* size, PartialView Cheating Degree *p*, is fixed in this simulation groups.



(a) Mean *InView* Size of Cheat



(b) Mean *PartialView Size* of Cheat



(c) Cheat Group Benefit Degree

Fig. 2. The Subscription Cheating Degree Impact.

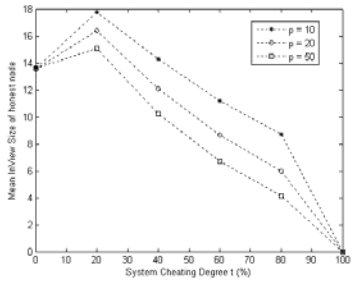
Fig. 2 (c) shows the Cheat Group Benefit Degree $\theta(n, s, p, t)$ against *s* and *t* when $p = 20$ and $n = 50000$. We find that by the method of increasing the Subscription Cheating Degree *s*, cheat group always can get benefit from the cheating behaviors, because $\theta(n, s, p, t)$ is always positive. And there are two key points in the Figure: one is at $t = 20\%$, cheat group gets the maximum benefit. When $t > 20\%$, bigger *t* is, less benefit the cheat group gains. As mentioned above, this is because when there are more cheats, the probability of cheat behavior between cheats will go up, and the benefit cheat group gains becomes less. The other key point is at $t = 100\%$, all the nodes in the system are cheats. Then they cannot gain benefits from any other nodes. As a result, this system gets a self-balance automatically.

As a whole, we conclude that cheat group can get benefit by adjusting the Subscription Cheating Degree *s*, which influences the cheat *InView* size directly. Bigger the *s* is, more the *InView* size of cheat will increase. From Fig. 2 (a) and (c), we also know that the benefit cheat group gains and the System Cheating Degree *t* do

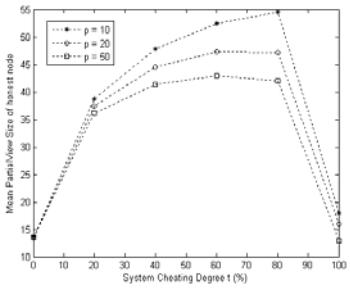
not direct ratio. When *n*, *s*, *p* are fixed, there is a value for *t* ($0\% < t < 100\%$) that makes cheat group have a maximum benefit, $\max\{\theta(n, s, p, t)\}$.

4.2 PartialView Cheating Degree Impact

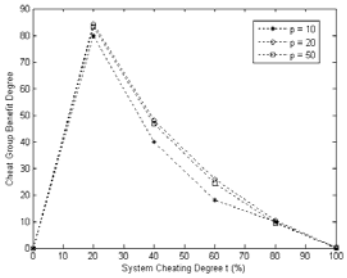
The PartialView Cheating Degree *p* represents the degree which cheats magnify the *PartialView* size to. This parameter influences node acceptance probability to the subscription requests. We analyze the influence of parameter *p* brings to the gossip-based protocol system from honest group point of view. The value of honest group *InView* size, *PartialView* Size against *p* and *t* when $s = 10$ are illustrated in Fig. 3 (a) and (b). Then the representative parameter Cheat Group Benefit Degree $\theta(n, s, p, t)$ is also analyzed in Fig. 3 (c).



(a) Mean *InView* Size of honest node



(b) Mean *PartialView Size* of honest node



(c) Cheat Group Benefit Degree

Fig. 3. The PartialView Cheating Degree Impact.

From Fig. 3 (a), we find that when p is fixed, the mean *InView* size of honest node first increases when $0 \le t \le 20\%$, then decreases as t grows. This is because that cheat sends more than one subscription requests during joining group ($s = 10$), and increases its *PartialView* size to decrease the probability being a provider for others ($p > 1$). In this circumstance, honest nodes are more likely to be providers, so its *PartialView* size will become much bigger than normal. Then when an honest node is going to join the group and sends a new subscription request to an arbitrary member, likely an honest one. According to the gossip-based protocol, the request will be replicated and forwarded to all the members in the receiver's own *PartialView*. For the *PartialView* size of honest node is larger than usual under this situation, the subscription request will be forwarded more times. As a result, the *InView* size of the new coming honest node increases (see Fig. 3 (a),when $0 \le t \le 20\%$). When t grows bigger, the mean *InView* size of honest node goes down. The reason is that when most of nodes are cheats who are prone to redirect the requests, the number of nodes willing to be providers becomes less. The subscription request is more likely to be discarded, so the honest node *InView* size decreases. In Fig. 3 (a), we can see that with a fixed t, the mean *InView* size of honest node decreases as p grows up, which is contrary to the effect of Subscription Cheating Degree s makes. The reason is that bigger p is, more likely node redirects the request.

Fig. 3 (b) shows the mean *PartialView* size of honest node against parameter p. We can see that when $0\% < t < 80\%$, the mean PartialView size of honest node goes up as t increases. When $t \ge 80\%$, the value decreases. This can be explained that when there are more cheats in the system, the probability of cheating behavior between cheats will increase, so the damage the honest node suffers goes down.

In Fig. 3 (c), Cheat Group Benefit Degree $\theta(n, s, p, t)$ against p and t is illustrated. We find that cheats also gain benefit from this cheating behavior. With a fixed t, bigger p is, more benefit cheats group gains. The reason is that when p grows, the probability of cheat being provider decreases, then its *PartialView* size goes down. Also we can see that when $t = 20\%$, cheat group gains maximal benefit, which is similar to Fig. 2 (c). This is also because the probability of cheating behavior between cheats will increase as more cheats in the system.

As Fig. 3 (a) to (c) illustrated, PartialView Cheating Degree p also influences gossip-based ALM protocol greatly. Bigger p is, more benefit cheats group gains, and more damage honest nodes suffers. Therefore, cheats always have the motivation to cheat more, so as to benefit more. Like parameter s impact, System Cheating Degree t and Cheat Group Benefit Degree $\theta(n, s, p, t)$ also have a delicate relationship. There exists a given value for t ($0\% < t < 100\%$) that makes cheat group get a maximum benefit.

5 Conclusion

In this paper, the impact of node cheating on the performance of gossip-based protocol system is analyzed. We find that there exists the severe potential trouble of trust on the gossip-based system. Nodes can make use of some simple cheating

strategies to improve its performance, and lead more burdens to honest nodes. That also jeopardizes scalability of the gossip-based protocol. We establish a cheating model to analyze cheating behaviors impact, and find that node cheating has considerably negative impact on the gossip-based protocol system. Therefore, node cheating should be paid much more attention during gossip-based protocol design.

As future work, we will try to design some cheat detection or prevention techniques to discover the cheating behaviors, avoid the negative impact of node cheating.

Acknowledgments. This research is supported by NCET-04-0079 (Program for New Century Excellent Talents in University, China).

References

1. K.P. Birman. The process Group Approach to Reliable Distributed Computing. *Communications of the ACM*, 36(12): 37-53, December 1993.
2. S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transaction on Computer Systems*, 8(2): 85-110, May 1990.
3. L. Mathy, N. Blundell, V. Roca, and A. EI-Sayed. Impact of Simple Cheating in Application-Level Multicast. In *the Proceedings of IEEE INFOCOM 2004*, March 2004.
4. D. Pendarakis, S. Shi, D. Verma and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, March 2001.
5. S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, August 2002.
6. B. Zhang, S. Jamin, L. Zhang. Host Multicast: A Framework for Delivering Multicast to End Users. In *Proceedings of IEEE INFOCOM 2002*, June 2002.
7. K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal Multicast. *ACM Transaction on Computer Systems*, 17(2): 41-88, May 1999.
8. A.J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-Peer Membership Management for Gossip-Based Protocols. *IEEE Transactions on Computers*, 52 (2), February 2003.
9. Q. Sun and D.C. Sturman. A Gossip-Based Reliable Multicast for Large-Scale High-Throughput Applications. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN2000)*, July 2000.
10. X. Zhang, J. Liu, B. Li, and T.-S.P. Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Live Media Streaming. In *Proceedings of IEEE INFOCOM2005*, March 2005.
11. A.-M. Kermarrec, L. Massoulié and A.J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3), March 2003.
12. D. Li, Y. Cui, K. Xu, and J. P. Wu. Impact of Receiver Cheating on the Stability of ALM Tree. In *Proceedings of GLOBECOM 2005*, November 2005.
13. P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN2001)*, 2001.