Cheating Avoidance Algorithms for an Overlay Multicast protocol

Ayman EL-SAYED, IEEE Member
Computer Science & Engineering Dept.,
Faculty of Electronic Engineering,
Menoufiya University, EGYPT
Site: http://ayman.elsayed.free.fr/
e-mail: ayman.elsayed@free.fr

***Abstract*— In this paper we improve the performance of an overlay multicast or End-system Multicast protocol (ESM), ESM is a multicast protocol where everything is being controlled by a single host called Rendez-vous point (RP). RP is responsible of overlay creation. Each group member informs RP by the metrics between itself and the other group members. Some of group members may be cheats. A cheat informs RP that it has the minimum delay to the source and the infinity delay to the other group members. The overlay multicast is not efficient due to the wrong metrics between group members because of these cheats. So, we propose algorithms to detect the cheat members and create an overlay multicast having performance like the overlay multicast without cheating.**

***I*ndex Terms — Multicast protocol, Ens-system Multicast, Application-level Multicast, Cheating.**

I. INTRODUCTION

End-system Multicast (ESM) [1] protocols creates an overlay depending on the metrics between the group members. Each group member collect the metrics between itself and the other group members. All metrics are collected either on a single node or on distributed nodes.

The End-System Multicast (ESM) can be classified into two main categories: (i) tree first approaches, where an overlay tree is constructed on the physical network as in [2], (ii) Mesh first approaches, where a mesh is constructed on the physical network and then a tree is created on the constructed mesh. The mesh first approach is classified into three categories: (a) a distributed protocols like NARADA [3], (b) centralized protocol like ESM [1] and Host-based Multicast (HBM) [4], (c) semi-centralized as described in [5], [6].

In [7], a highly scalable locating algorithm is proposed to gradually direct newcomers to their a set of their closest nodes without inducing high overhead. On the basis of this locating process, they build a robust and scalable topology-aware clustered hierarchical overlay scheme to support large scale multicast applications.

If we consider the very popular round-trip time (RTT) distance measurements used in end-system multicast protocols, a cheat could delay a probe received from another receiver to artificially increase their measured distance in order to try and reduce its replication burden (since the further away another receiver is, the more likely that receiver will be connected to another (closer) node). For scalability reasons, most of the existing end-system multicast protocols require that each node measures its distance to other nodes in the overlay tree and reports these distance measurements to other nodes and/or uses these for decision making. A cheat can therefore lie outright about its distance measurements, in order to try and improve its position in the tree [8].

It is important to note that the cheats considered in this paper do not attempt to disrupt the flow of data along the overlay tree or even to break the protocol used to build the tree, they simply try and improve their position in the tree. Some end-system multicast protocols are described and ad-hoc multicast protocols with the cheating in [9] and [10] respectively. Member (i.e. receiver) cheating may transform the multicast tree, and lead to its instability. In [11], the authors establish the cheating model of end-system multicast receivers and analyze the stability of overlay tree topology when receiver cheating occurs.

The remainder of the paper is organized as follows: we describe the impact of cheats in section II, the algorithm to detect the cheating in section III, propose

algorithms to know the selfish (cheat) members and cancel the effect of cheating in section IV, and the results are discussed in section V. Finally, we conclude the paper in section VI.

II. Impact of Cheats

Since the hosts in the overlay multicast are selfish, without a proper payment scheme, they may not forward others messages or they may try to cheat the system, if the cheating can maximize their welfare. In particular, a selfish node can exhibit one of the three selfish actions:

- After receiving a message, the node saves a receipt but does not forward the message;
- The node has received a message but does not report the receipt;
- The node does not receive a message but falsely claims that it has received the message.

Note that any of the selfish actions above can be further complicated by collusion of two or more nodes. We next progressively determine the requirements on our system in order to prevent the above actions.

The important point here is that there is an opportunity for receivers to try and improve their position on the overlay tree by manipulating distance measurements, in order to be positioned closer to the data source while limiting, to a minimum, their replication burden.

In another word, we show an example. There are 20 hosts started with identifiers from ”2” to ”21” and a source having identifier ”1” and each host has the maximum number of neighbors of 6. Firistly, we suppose that there is no selfish hosts. Figure 1 shows an overlay network without selfish hosts, where the dashed circule is the source host, the bold circule is noncheat host and the thin bold line is the link between two hosts.

Now, we suppose that there are selfish hosts (both 20% and 75% of hosts are cheat hosts) shown in Figure 2 and 3 respectively. Selfish hosts make both the delay metric between itself and the source be either zero or right value (real). Also they make the delay between itself and the other hosts be the highest (infinity), by adding 10 sec to the real dealy (i.e. honor↔cheat = real delay+10sec). In the case of two selfish hosts, adding 10sec for each one (i.e. cheat↔cheat = real delay+20sec).

In case of small number of cheats (20% of cheats) as shown in Figure 2, the selfish hosts are shown as

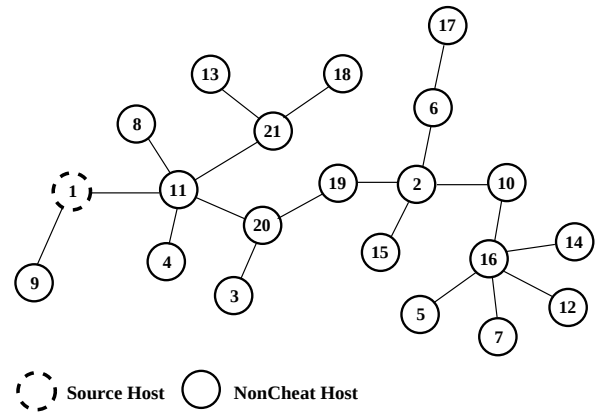


Fig. 1. An overlay multicast without cheats

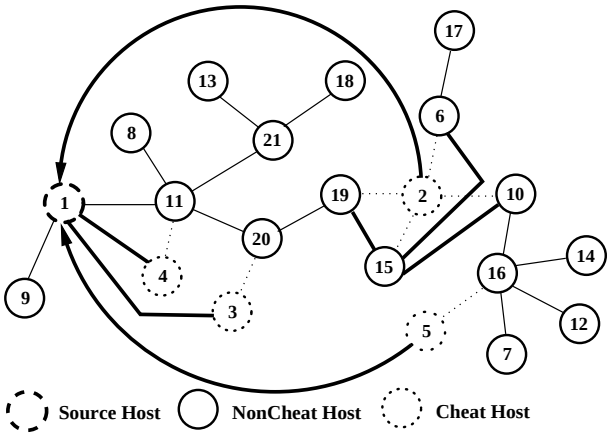
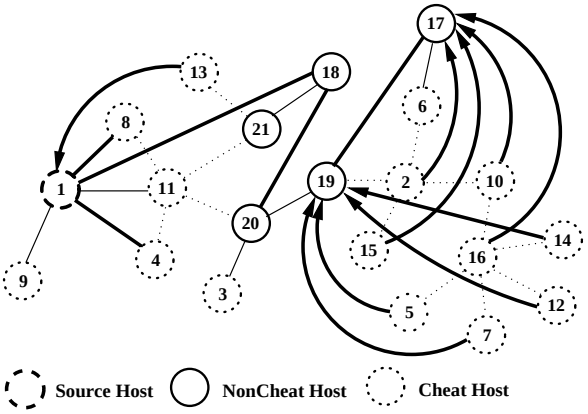


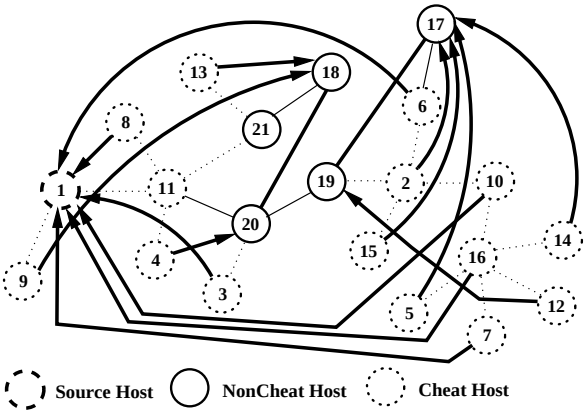
Fig. 2. An overlay multicast with 20% of cheats, with maximum number of neighbours=6

dotted circule (i.e.hosts: 2, 3, 4, and 5). All selfish hosts are directly attached to the source. But the number of cheats is less than the maximum number of neighbours (fanout). We note that all noncheat hosts are connected to the source directly not via selfish hosts. We show that the selfish host ”2” for example, improves its position in the overlay. It connected to the source host instead of connected to the hosts 6, 10, 15, and 19. So it satisfies the objective of cheating. Also, we note that the selfish hosts are connected to the source and they has no childern hosts. There is no big changing of overlay network but the source has maximum number of neighbors (i.e 6).

In case of most hosts being cheat, there are two issues:

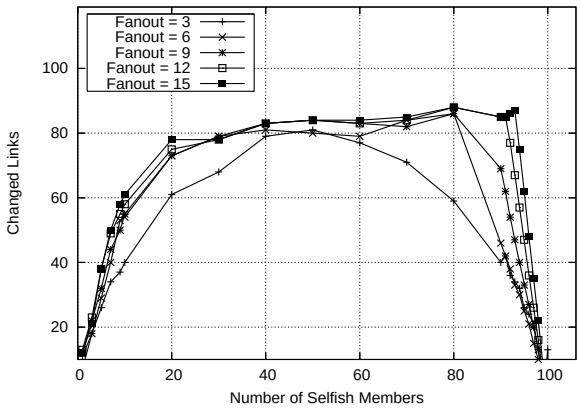


(a) Source↔cheat delay=real

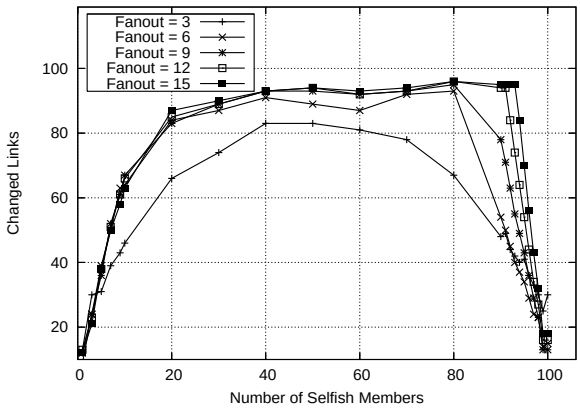


(b) Source↔cheat delay=zero

Fig. 3. Overlay multicast with 75 % of cheats, with maximum number of neighbours=6



(a) Source-to-Cheat Real Delay



(b) Source-to-Cheat Zero Delay

Fig. 4. The effect of cheating on overlay Topology

- **Source-to-Cheat Real Delay:** cheat↔source = real delay, it means that the selfish member let its delay to the source as it is and increase its delay to the other members by a factor, as shown in Figure 3-a.
- **Source-to-Cheat Zero Delay:** cheat↔source = zero, it means that the selfish member set its delay to the source to zero and increase its delay to the other members by a factor,as shown in Figure 3-b.

In general, we describe the effect of cheating for the previous two issues that are shown in Figure 4. This figure depicts the number of changed links because of cheating versus number of selfish members with varying number of neighbors such as 3, 6, 9, 12, and 15, where number of all members = 100 (both selfish and honor) members.

We note that increase in number of neighbors, increase the number of changed links because with increasing the number of neighbors takes chance for selfish members to connect the source. When number of selfish members less than 20% of all members, the number of changed links approximatly linear increase because the many metrics among members equal real values and the other metrics are increased by a cheated time delay or double a cheated delay. When number of selfish members is more than 90%, the number of changed links linear decreases, and the metrics having real values is very small. The other metrics increased by a fixed value, so the metrics are increased by the multiple of its real values. So, the effect of cheating becomes a big problem when the number of selfish members is more than the maximum number of

available neighbours.

III. Detect the Cheating

First, we suppose the distance (ie. time delay) between node i and node j as $D_{i\leftrightarrow j}$, the node i that has parent p is N_i^P, and the neighbour node j of node i is N_j^i. There is a cheating, if there are neighbors of the source (N_j^s) and they have a neighbor (i.e. the source) only except one of them that has two neighbors (i.e. the source and another member). This case is appeared when the number of selfish members is more than the maximum number of available neighbors in the overlay topology. It is important that we detect there is cheating (e.g. at least one selfish member) to apply our cheat avoidance algorithms.

A. Source-to-Cheat Real Delay

In order to detect the selfish hosts in this case, we use the following: For each neighbour (N_j^S) of source (S), if ($D_{S\leftrightarrow N_j^S}$ =real value && there is no N_i^j) then N_j^S may be cheat (i.e. selfish node).

In this case, we found the source is connected to either selfish nodes and honor nodes (i.e. *state (I)*) or selfish nodes only (i.e. *state (II)*), that is depending upon the number of selfish hosts and the delays between the source and all nodes. In this case we can say that *there is a cheating*.

B. Source-to-Cheat Zero Delay

In order to detect the selfish hosts in this case, we use the following: For each neighbour (N_j^S) of source (S) if ($D_{S\leftrightarrow N_j^S}$ =Zero or real value && $D_{N_j^S\leftrightarrow N_i^j}$ = real value && N_i^j is one neighbor only) then N_j^S may be cheat. In this case we can say that *there is a cheating*.

IV. Avoid Cheats

A. Parameters Definations

We suppose the maximum time delay (D_{max}), minimum time delay (D_{min}), and the average time delay (D_{avg}) = $\frac{D_{max}+D_{min}}{2}$, the minimum time delay between a selfish member and honour member ($D_{MinOneCheat}$), we can say that there is a selfish member (N_i) where N_j is a honor member, if the time delay ($D_{i\leftrightarrow j}$) is more than $D_{MinOneCheat}$. Also, we can say that there are two selfish members if the minimum time delay among them more than the minimum time delay among two selfish members ($D_{MinTwoCheat}$). But the question here how to obtain both $D_{MinOneCheat}$ and $D_{MinTwoCheat}$? Because selfish member increases its time delay between itself and other member except the source, then the time delay among two selfish members is increased two times but that delay between a selfish and an honor member is increased one time only. These parameters are shown in Figure 5. We can obtain $D_{MinOneCheat}$ and $D_{MinTwoCheat}$ as following:

$D_{MinOneCheat} = \frac{D_{min}+D_{avg}}{2}$.

$D_{MinTwoCheat} = \frac{D_{avg}+D_{max}}{2}$.

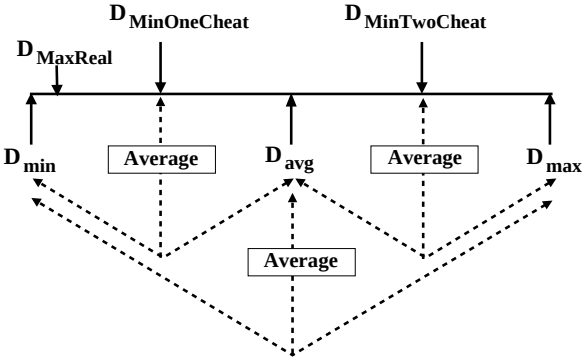


Fig. 5. Detect Cheats Boundaries

B. Detect/Know Selfish Members

After detecting the cheating, we can count and Know all slefish members by the following algorithm:

- D_{min} = get minimum of all metrics between each pair of members.
- D_{max} = get maximum of all metrics between each pair of members.
- D_{avg} = $\frac{D_{max}+D_{min}}{2}$.
- $D_{MinOneCheat}$ = $\frac{D_{min}+D_{avg}}{2}$.
- $D_{MinTwoCheat}$ = $\frac{D_{avg}+D_{max}}{2}$.
- Set counter (c = 0).
- Choose any member except the source, say N_2 for example.
- Repeat for each j = 3...N, where N: Number of all members:
  - if $D_{2\leftrightarrow j} > D_{MinTwoCheat}$), then both N_2 and N_j are selfish members then add them to a selfish members group, with taking into

account no repeated member in this selfish members group.
  - if ($D_{2\leftrightarrow j} > D_{MinOneCheat}$ and $D_{2\leftrightarrow j} < D_{MinTwoCheat}$), then either N_2 or N_j is a selfish member, if N_2 is added before in selfish members group, then do nothing, else add both N_2 and N_j as a member-pair unit in doubted members group.
- Check each member-pair unit in doubted group:
  - if there is a repeated member in member-pair unit, then add another member in member-pair unit to selfish members group and delete these member-pair unit.
  - if a member in member-pair unit occurs in the selfish member group, delete this member-pair unit, repeat this step until doubted group is empty.

Let the maximum real time (i.e. among two honor members) is $D_{MaxReal}$ as shown in Figure 5, we can get $D_{MaxReal}$ by getting the maximum time delay among honor members only, as following:

$D_{MaxReal}$ = MAX{all honor members-pair time delay}

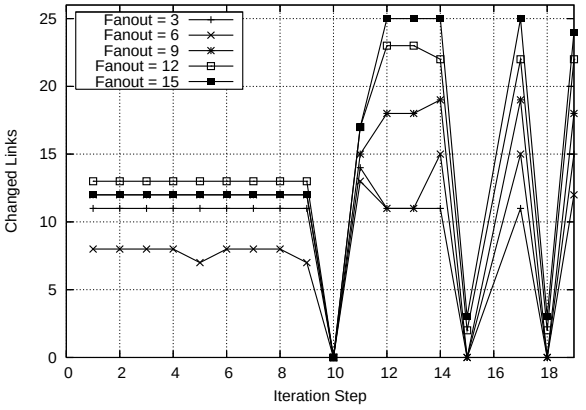
C. Cancel Cheating Effect: Adapting Metrics

After getting $D_{MaxReal}$ among honors members only, we note that the time delay among two members that are either one selfish member and an honor member or two selfish members is more than $D_{MaxReal}$. These metric can be decremented by a step called *iteration step* (Stp_{dec}), till the metric becomes either less than or equal to $D_{MaxReal}$. The question here, what is the value of Stp_{dec}? Sure, the value is related to $D_{MaxReal}$ but what is the value of Stp_{dec}? Here in this paper, we put Stp_{dec} as some values less than $D_{MaxReal}$ and some values more than $D_{MaxReal}$, as described in the following sections.

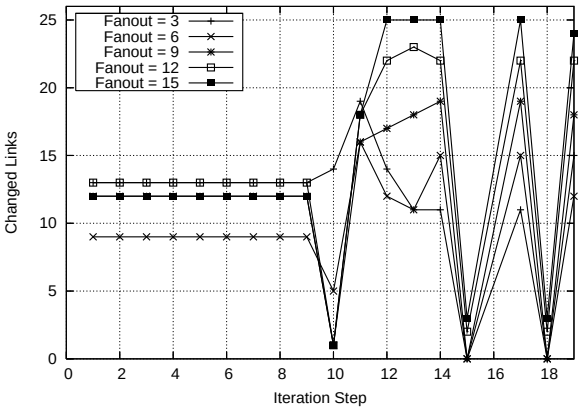
V. DISCUSSION AND RESULTS

We suppose the number of all members (N) is 100 members including one source that having identifier of (N_1), and the real delays among each two members-pair is in the range from 1 ms to 10 ms.

After detecting the cheating, we can apply our algorithms to modify the metrics in the cases: a selfish member only, and selfish members of 20%, 50%, 80%, and 100% of all members, as following: **A cheat member:** Figure 6 shows the number of changed links versus the iteration step, with one member cheats. In case of both Source-to-Cheat Real Delay (Figure 6-a) and Source-to-Cheat Zero Delay (Figure 6-b), when iteration step is less than 10, the number of changed links is approximatly constant in range from 10% to 15% of all links. When iteration step more than 10, the number of changed links is vibrated from zero to 25% of all links. Finaly, we note that at iteration step = 10 that is equal to $D_{MaxReal}$, we find there is no changing in the overlay in case of Source-to-Cheat Real Delay and the changed links is less than 15% of all links in the case of Source-to-Cheat Zero Delay.



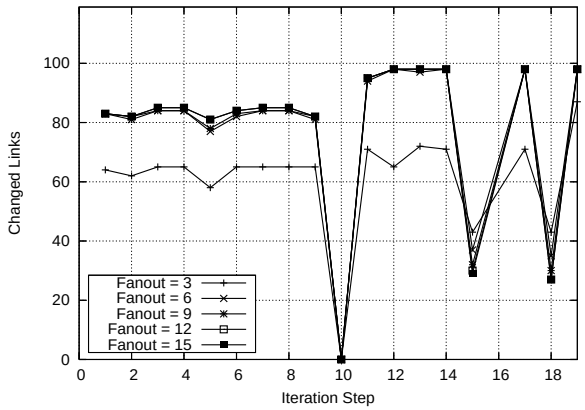
(a) Case I: Real time between source and selfish member



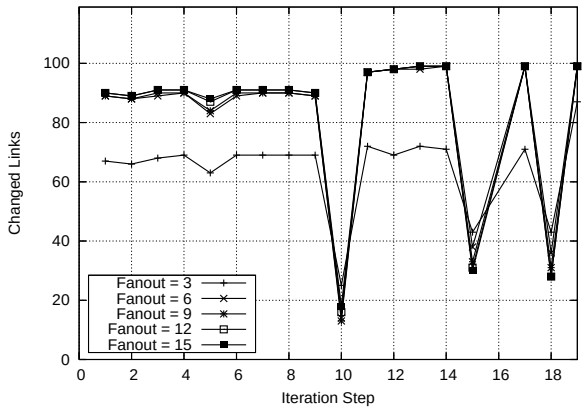
(b) Source-to-Cheat Zero Delay

Fig. 6. Changed Links versus Iteration step, with $D_{MaxReal}$= 10ms and one member only cheats

20% cheat members: Figure 7 shows the number of changed links versus the iteration step, with cheats = 20%. In case of Source-to-Cheat Real Delay (Figure 7-a), when iteration step is less than 10, the number

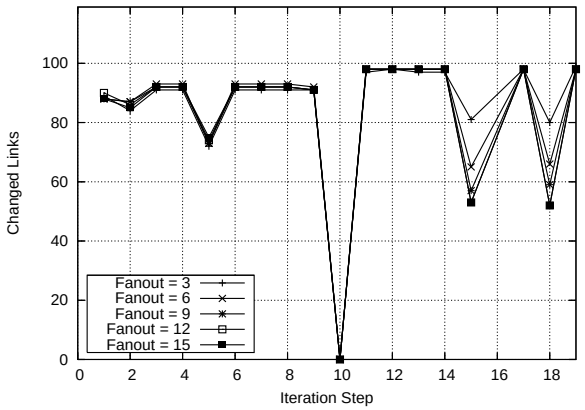


(a) Source-to-Cheat Real Delay

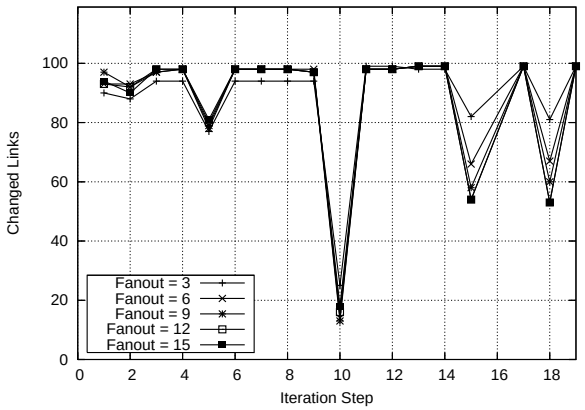


(b) Source-to-Cheat Zero Delay

Fig. 7. Changed Links versus Iteration step, with $D_{MaxReal}$= 10ms and Cheats = 20%.



(a) Source-to-Cheat Real Delay



(b) Source-to-Cheat Zero Delay

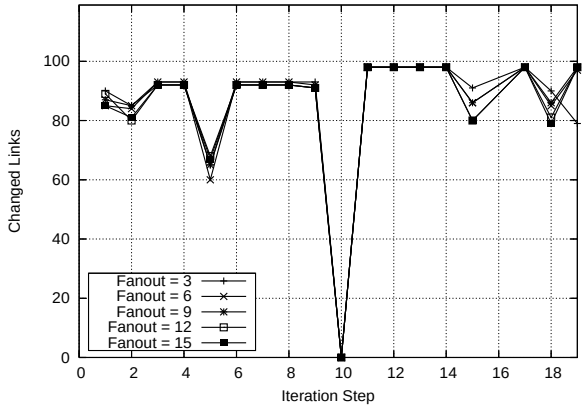
Fig. 8. Changed Links versus Iteration step, with $D_{MaxReal}$= 10ms and Cheats = 50%.

of changed links is approximatly constant and when iteration step more than 10, the number of changed links is vibrated. Finaly, we note that at iteration step = 10 that is equal to $D_{MaxReal}$, we find there is no changing in the overlay when 20% of all members cheat.

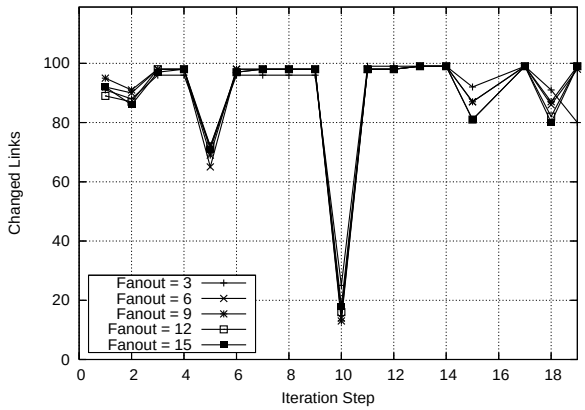
In case of Source-to-Cheat Zero Delay (Figure 7-b), we note that at iteration step = $D_{MaxReal}$, the number of changed links is the smallest. It means, we can't get the real delay between the source and the selfish because these delays become zero. But our algorithm also modify the metric to get an overlay more near to overlay without cheating. **50% cheat members:** Figure 8 shows the number of changed links versus the iteration step, with cheats = 50%. Similar results as in Figure 7 at iteration step equals $D_{MaxReal}$ (10 ms).

80% cheat members: Figure 9 shows the number of changed links versus the iteration step, with cheats = 80%. Similar results as in Figure 7 at iteration step equals $D_{MaxReal}$ (10 ms).

100% cheat members: Figure 10 shows the number of changed links versus the iteration step, with cheat = all members. Similar results as in Figure 7 at iteration step equals $D_{MaxReal}$ (10 ms). We note that our algorithm success to modify the metrics to obtain the overlay topology like that of no cheats occur in the case of **Source-to-Cheat Real Delay** as shown in Figures 6-a, 7-a, 8-a, 9-a, and 10-a. Our proposed algorithms completly cancel the effect of cheating in the case of **Source-to-Cheat Real Delay**, it means that there is no change in the overlay topology when there

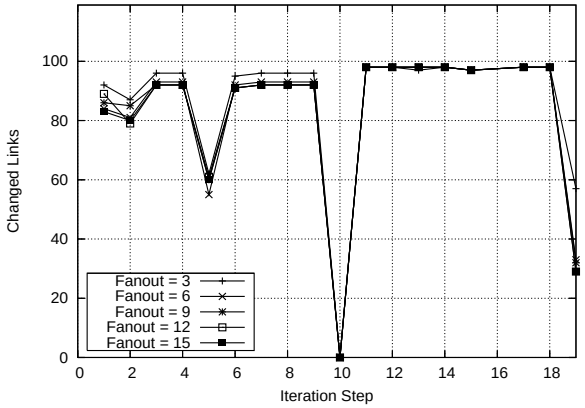


(a) Source-to-Cheat Real Delay

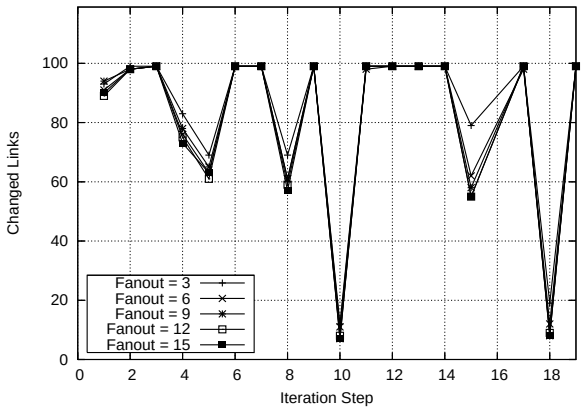


(b) Source-to-Cheat Zero Delay

Fig. 9. Changed Links versus Iteration step, with $D_{MaxReal}$= 10ms and Cheats = 80%.



(a) Source-to-Cheat Real Delay



(b) Source-to-Cheat Zero Delay

Fig. 10. Changed Links versus Iteration step, with $D_{MaxReal}$= 10ms and All Members Cheat.

is a selfish member or more. But in the case of **Source-to-Cheat Zero Delay**, our algorithm can decreases the number of changed links, less than 25% of all overlay links as shown in Figures 6-b, 7-b, 8-b, 9-b, and 10-b. Before it is more than 80% as shown in Figure 4. The summary is shown in Figure 11. We note that our algorithms decrease the number of changed links to be equal zero or less than 25% of all links. So our algorithm can cancel the effect of cheating in both issues.

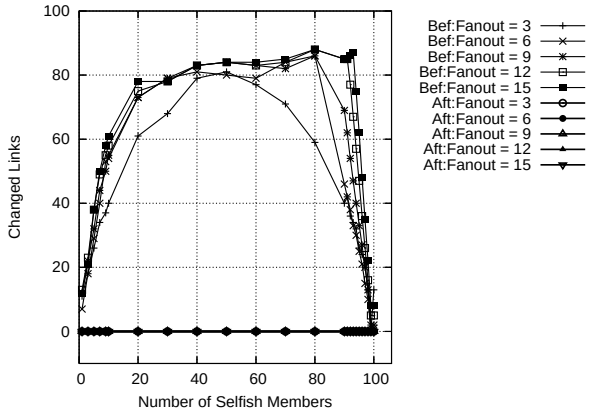
VI. Conclusion

In this paper, we have studied the impact of cheating on the performance of application-level multicast overlay trees. We have shown that the cheating always have negative impact, either on the performance of the tree as perceived by its nodes (both cheats and honest receivers), or on the underlying physical network, or on both. We proposed an algorithm to detect/know the selfish member and another algorithm to cancel the effect of cheating on the metric values. these algorithm can obtain the metrics by that the overlay topology is the same like that without cheating.

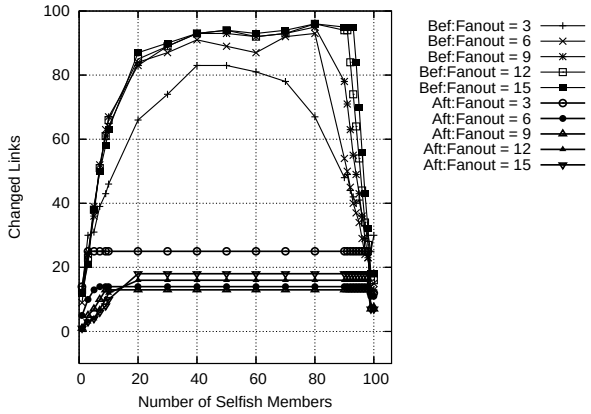
References

[1] Anirban Chakrabarti and Govindarasu Manimaran, "A case for mesh-tree-interaction in end system multicasting," in *NETWORKING 2004, LNCS 3042, pp. 186-199*, November 2004.

[2] Laurent Mathy, Roberto Canonico, and David Hutchison, "An overlay tree building control protocol," in *proceeding of the third International COST264 Workshop, Networked Group Communication (NGC 2001), London, UK, pp. 76-87*, Nov. 2001.



(a) Source-to-Cheat Real Delay



(b) Source-to-Cheat Zero Delay

Fig. 11. Changed Links versus cheats, with $Stp_{dec} = D_{MaxReal}$

[3] Yang hua Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS, pp. 1-12*, June 2000.

[4] Vincent Roca and Ayman El-sayed, "A host-based multicast (hbm) solution for group communications," in *First IEEE International Conference on Networking (ICN'01), Colmar, France, pp. 610-619*, July 2001, pp. 610–619.

[5] Ayman El-sayed, "Semi-centralized approach for end-system multicast protocol," in *Menofia Journal of faculty of Electronic Engineering Research (MJEER), 15-2*, july 2005.

[6] Ayman EL-SAYED, "A new approach for centralized end-system multicast protocol," in *International Journal of Information Acquisition (IJIA), Vol. 3, No. 1*, March 2006.

[7] Mohamed Ali Dali Kaafar, Thierry Turletti, and Walid Dabbous, "A locating-first approach for scalable overlay multicast," in *14th IEEE International Workshop on Quality of Service, IWQoS2006.*, June 2006.

[8] Mike Afergan and Rahul Sami, "Repeated-game modeling of multicast overlays," in *IEEE INFOCOM 2006, Barcelona, Spain*, Apr. 2006.

[9] Laurent Mathy, Nick Blundell, Ayman EL-SAYED, and Vincent ROCA, "Impact of simple cheating in application-level multicast," in *IEEE INFOCOM*, July 2004.

[10] Sheng Zhong, Jiang Chen, and Yang Richard Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," in *IEEE INFOCOM*, July 2003.

[11] D. Li, Yong Cui, Ke Xu, and Jianping Wu, "Impact of receiver cheating on the stability of alm tree," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE, Vol.2, Iss.*, Dec. 2005.