This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Truthful Streaming in Selfish DONet

Dan Li, Jianping Wu, Yong Cui

Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
Email: lidan@csnet1.cs.tsinghua.edu.cn, jianping@cernet.edu.cn, cy@csnet1.cs.tsinghua.edu.cn

Abstract—**Data-driven Overlay Network (DONet) is especially suitable for live stream because it can tolerant node dynamics well. However, optimal streaming demands the cooperation of individual nodes. If selfish nodes in DONet cheat about their buffer maps to reduce the forwarding burden, the overall streaming quality might be negatively affected. To defend this kind of cheating behavior, we design a trustworthy service-differentiation based incentive mechanism with low complexity in this paper. The mechanism is composed of the service-differentiation algorithm and the contribution-evaluation algorithm. Compared with other studies in this area, the primary characteristic of our mechanism lies in two aspects. Firstly, the contribution of each node is evaluated considering the characteristic of live stream, not just by the transferring bytes. Secondly, the potential cheating behavior of overlay nodes during the fulfillment of incentive algorithms can be defended, which is usually not considered by other studies.**

I. Introduction

With the rapid growth of Internet, multimedia applications, especially live stream, have been used more and more [1]. In live stream, a large number of users are interested in the real-time data from a common source. Compared to other applications, live stream demands higher network bandwidth as well as node forwarding capacity. Given the multi-receiver nature of live stream, multicast is the ideal supporting technology. Currently, there are two kinds of multicast technologies. One is realized in the network layer, named as IP multicast [2]; and the other is realized in the application layer, named as overlay multicast [3~14].

IP multicast builds the data structure on routers, which is a tree, and thus achieves high scalability and high efficiency. However, IP multicast changes the "unicast" principle of traditional Internet, and a lot of problems in it, such as member management, congestion control, and pricing model, have not been solved well yet. All these lead to the difficulty of deploying IP multicast in Internet scale.

Overlay multicast is subsequently proposed, which constructs the data structure in the application layer. Compared with IP multicast, overlay multicast is lower in efficiency, but much more deployable and flexible. Currently, there are two kinds of overlay multicast, namely, tree-based overlay multicast [3~9] and mesh-based overlay multicast [10~14]. Like in IP multicast, the data structure in tree-based overlay multicast is also a tree, and data is propagated along the tree after its establishment. However, overlay nodes are unstable. There is a high frequency of node join, node leave, and node crash in overlay network. Thus, the application-layer multicast tree might change from time to time, which will bring negative impact on live stream applications that have stringent demands on the streaming continuity.

Mesh-based overlay multicast is considered as a better choice to support live stream. In mesh-based overlay multicast, the data structure is no longer a tree, but a mesh. Data-driven Overlay Network (DONet) is a representative of this kind of protocols [10]. In DONet, the stream propagated in the overlay network is divided into multiple segments. Each node maintains a number of segments, and exchanges the buffer map of available segments with partner nodes. After learning the buffer maps of partner nodes, each node requests a certain segment from a suitable partner node that holds the segment. If a node receives segment requests from partner nodes, it replies to the requests by forwarding the corresponding segments within its outgoing bandwidth. Therefore, the leave or crash of a single node will not bring too much impact on other nodes.

However, overlay nodes are not only unstable, but also selfish and strategic. Selfish nodes in DONet might cheat about their buffer maps to reduce the forwarding burden to other nodes, which is to be detailed in Section III. In such non-cooperative scenarios, some nodes cannot have the actual information to request segments, and thus the streaming quality in DONet may not be optimized.

To defend buffer map cheating and maximize the streaming quality in selfish DONet, we design a trustworthy service-differentiation based incentive mechanism with low complexity in this paper. Each node is bound with a contribution index, recording its forwarding contributions to other nodes. As a reward, nodes with higher contribution indices will obtain better services when they request data from other nodes. The incentive mechanism is composed of the service-differentiation algorithm and the contribution-evaluation algorithm. Compared with other studies in this area, the primary characteristic of our incentive algorithms lies in two aspects. Firstly, the contribution of each node is evaluated considering the characteristic of live stream, not just by the transferring bytes. Secondly, the potential cheating behaviors of overlay nodes during the fulfillment of incentive algorithms can be defended, which is usually not considered by other studies.

II. Related Work

Because of the difficulty of deploying IP multicast in Internet scale, researchers turn to overlay multicast to support live stream applications. The overlay multicast protocols

* This work was supported by the National Natural Science Foundation of China (No. 60403035), the Hi-Tech Research and Development Program of China (No. 2006AA01Z205), and the National Basic Research Program of China (No. 2003CB314801).

1-4244-0353-7/07/$25.00 ©2007 IEEE

proposed currently can be roughly classified into two categories, namely, tree-based overlay multicast and mesh-based overlay multicast.

In tree-based overlay multicast, each node selects a long-time parent from other participating nodes to receive stream data. The parent/children relationships among all nodes compose the data structure, i.e., the multicast tree. Once the multicast tree is established, data is propagated along the tree and there is no additional control overhead. Protocols belonging to this category of overlay multicast include NARADA [3], NICE [4], ZIGZAG [5], Scattercast [6], Yoid [7], Host Multicast [8], ALMI [9], and etc.

In mesh-based overlay multicast, there is no explicit parent/ children relationship. The data structure to propagate the stream is a mesh. Therefore, it can tolerant node dynamics well and is especially suitable to support live stream applications. Representatives of this kind of protocols include gossip-based protocols and DONet. In gossip-based protocols [11~14], each node forwards available data to a set of randomly selected nodes. But in DONet [10], each node maintains several partner nodes, and data is transmitted among partner nodes, eventually to the whole overlay network. Compared to gossip-based protocols, the advantage of DONet is that data is flowing in a request-reply way, thus there is no redundant data consuming the precious network bandwidth.

An important characteristic of overlay network is that the overlay nodes are all selfish and strategic. The selfish overlay nodes might cheat about their private information to obtain higher interests, which could affect the performance of the overall system. One possible solution to this problem is to design monetary-payment based mechanisms to encourage selfish nodes to tell their true private information [15~19]. In this method, each node pays to the society for resource consumption and gets payment in return for resource contribution. Since payment evaluation policy and imaginary currency are needed in such mechanisms, considerable burden is added to the overlay networks. Another problem is who will fulfill the payment policy, which .is often not budget-balanced.

Another possible solution to the selfish cheating problem is service-differentiation based incentive mechanisms [20~23]. In this method, the historical contributions of all nodes are recorded, and nodes with higher contributions will obtain better services from the society. This method has been used in both overlay file-sharing applications and streaming applications. Buragohain et al. [20] propose a game theoretic framework to provide incentives in P2P system. In this framework, if a peer contributes more to the system, it earns a higher probability with which other peers reply to its requests. The peer contribution is quantified in terms of disk space shared per unit time. Nowak et al. [21] bring forward KaZaA, a score-based P2P system, which provides downloading priority to the users with high scores over those with low scores. Ma et al. [22] model the whole resource request and distribution process in P2P system as a competition game, and show that there is Nash equilibrium in the game. The file-sharing systems above usually only concern the availability of data. But it is not the case in streaming applications, because the streaming continuity is more important than just the data availability. Habib et al. [23] design a service-differentiation based mechanism to encourage cooperation in overlay streaming applications. Contributions of overlay nodes are recorded and nodes with higher contributions are rewarded with more flexibility in peer selection when they are downloading data. However, the authors do not consider that the selfish nodes might also cheat when fulfilling the incentive algorithms.

We design a service-differentiation based incentive mechanism in this paper to defend buffer map cheating in selfish DONet. Our mechanism is different from all above studies in that the contribution-evaluation algorithm considers the specific characteristic of live stream, and the potential cheating behavior of overlay nodes during the fulfillment of incentive algorithms are also defended.

III. Buffer Map Cheating in Selfish DONet

The stream propagated in DONet is divided into multiple segments with uniform length. A buffer map can represent the information of available segments on a node. Each node periodically exchanges its buffer map with partner nodes, and decides from which partner node to fetch a certain segment. If there are multiple partner nodes holding the same expected segment, various ways can be chosen to select the segment-providing node, for instance, the one with the shortest distance or the one with the highest outgoing bandwidth. The requests arrive at the requesting queue of the segment-providing node. When replying the requests, the segment-providing node selects some requests in the requesting queue and sends the corresponding segments within its outgoing bandwidth.

At first, we make some definitions and list the notations throughout this paper as Tab. I.

TABLE I. List of notations

N	Number of nodes in the overlay network
M	Number of partner nodes each node maintains
Q	Number of segments of the stream
$Pr(i,s)$	Preference index of request from node i for segment s
$Ct(i)$	Contribution index of node i
$Co(i,s,k)$	Contribution of node i after forwarding segment s to node k
$Ar(s,k)$	Arriving time of segment s on node k
$De(s,k)$	Playback time of segment s on node k
Tl	Longest tolerance time for a node to wait for a certain segment

Overlay nodes are all selfish agents. We discuss a potential selfish behavior of overlay nodes in DONet, that is, buffer map cheating. As shown in Fig. 1, the buffer map of node i is composed of segments 4, 5, 6, 7, 8, and 9. If node i exchanges the actual buffer map with partner nodes a and b, node a will request segments 4, 5, and 6 from it, and node b will request segments 5 and 6 from it. However, in order to reduce the forwarding burden, selfish node i could cheat about its buffer

map as segments 5, 7, and 9. Under this situation, node *a* and node *b* will request only segment 5 from node *i*, and they have to request the other demanded segments from other nodes, which are not so "good" as node *i*.

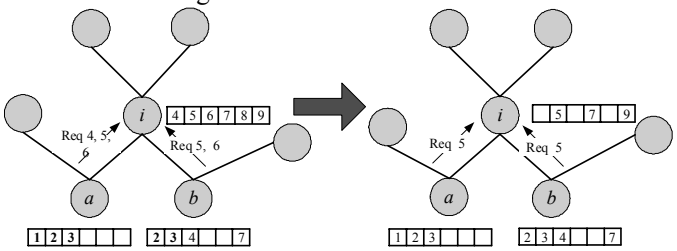

Fig. 1 Buffer map cheating of selfish node *i*

As hinted by Fig. 1, since the cheating node hides some of the available segments, it will not be requested by its partner nodes for these hidden segments. The forwarding burden of the cheating node will thus decrease. However, the partner nodes that originally expect to request the hidden segments from the cheating node will not be successful any more, and they have to request from secondarily best segment-providing nodes. As a result, their benefits will be reduced. From the view of the whole overlay network, the buffer map cheating will negatively impact the desired data flow, and the social benefit of the stream will not be optimized.

In literature [24], the authors studied the impact of buffer map cheating on the streaming quality in DONet by experiments, and found that buffer map cheating would bring considerable negative impact on the streaming in DONet. In order to obtain the optimal social benefit, we design an incentive mechanism to defend buffer map cheating in the following section.

IV. Design of Trustworthy Service-differentiation based Incentive Mechanism

The incentive mechanism we design to defend buffer map cheating in DONet is service-differentiation based. Our design goal is that each selfish and rational node will not cheat about its buffer map under the incentive mechanism, and that the potential cheating behavior of selfish nodes during the fulfillment of the incentive mechanism can also be defended.

In our mechanism, the historical forwarding contribution of each node *i* is recorded and is quantified as its contribution index, denoted by $Ct(i)$. To award the forwarding behaviors of overlay nodes, nodes with higher contribution indices receive better services. When the segment-providing node replies the requests in the requesting queue, the requests from the nodes with higher contribution indices will be more favored. It is assumed that the contribution indices of all nodes are maintained on the source node of the multicast session. The source node can be regarded as a trustworthy third-party, because its benefit is associated with the overall outcome of the multicast session, while not that of individual nodes. Each node can obtain the contribution indices of all nodes from the source node, either periodically, or request-driven.

The incentive mechanism we design is composed of two parts: the service-differentiation algorithm and the contribution-evaluation algorithm. The service-differentiation algorithm describes how for segment-providing nodes to differentiate the services to requests from nodes with different contribution indices, and the contribution-evaluation algorithm focuses on the quantification of the nodes' forwarding contributions.

A. Service-Differentiation Algorithm

The segment requests from different overlay nodes are assigned different preference indices. The preference index assigned to a request from node *i* for segment *s* is denoted as $Pr(i,s)$. After arriving at the segment-providing node, each segment request is assigned its preference index, and put into the requesting queue of the segment-providing node according to its preference index. When the segment-providing node replies the segment requests, it selects the requests with higher preference indices and sends the corresponding segments within its outgoing bandwidth.

To encourage selfish nodes to forward available segments, the requests from nodes with higher contribution indices are assigned higher preference indices, as illustrated by Fig. 2. As stated above, the contribution indices of the requesting nodes are obtained from the source node. The contribution-preference mapping function is arbitrary, only if the preference index is non-decreasing over the contribution index. An example of mapping function is shown in Eq. (1).

$$Pr(i,s) = Ct(i) \qquad (1)$$

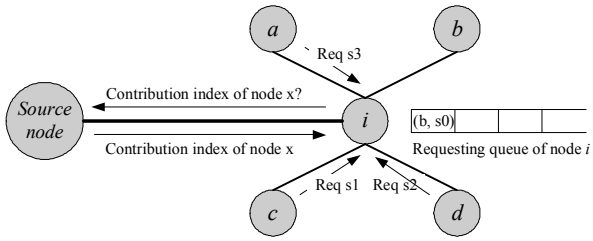

Fig. 2 Service differentiation on segment-providing node *i*

However, if the segment-providing node assigns the preference index of the request only considering the contribution index of the requesting node, there is a potential problem that the nodes with lower contribution indices may have the chance of "starvation", because of continuously new-coming segment requests from nodes with higher contribution indices. This is not in favor of stream propagation in the whole overlay network.

We take the sequence number into consideration to solve this problem. Since each node plays the stream from lower--sequence segments to higher-sequence segments, segments with lower sequence numbers are demanded more urgently in DONet. According to [25], lower-sequence favored streaming will decrease the playing delay. Therefore, when assigning the preference indices to the segment requests, the requests for lower-sequence segments are also favored.

After the improvement, the preference index of a request is non-decreasing over the contribution index of the requesting

node, and is non-increasing over the sequence number of the requesting segment. The example mapping function shown in Eq. (1) can be extended to Eq. (2).

$$Pr(i,s) = \frac{Ct(i)}{s+1} \qquad (2)$$

The service-differentiation algorithm on the segment-providing node can be illustrated in Tab. II.

TABLE II. Service-differentiation algorithm on the segment-providing node

```
    // processing an arriving segment request
1   void reqMsgProcess (reqMsg){
2     k←reqMsg.source
3     s←reqMsg.segment
        // obtaining the contribution index of node k from the
        // source node
4     Ct(k)←contReceiveFromSource(k)
      // preference index assignment
5     Pr(k,s) ← Ct(k)/(s+1)
      // insert the request into the requesting queue according to
      // its preference index, higher preference index nearer the
      // queue head
6     queueInPr(k, s, Pr(k, s))
7     return
8   }

   //service differentiation for requests in the requesting queue
1   void serviceDiff(){
2     while (true)
        // there is requests in the requesting queue
3       if !queueEmpty()
          // get the requests near the queue head within the
          // outgoing bandwidth
4         queueOut (&reqs)
          // reply the requests by sending the requesting
          // segments to the corresponding requesting nodes
5         segmentSend(reqs)
6       end if
7     end while
8     return
9   }
```

After introducing the service-differentiation based incentive mechanism, the benefit each node obtains from receiving stream data becomes related with its forwarding contribution. Higher forwarding contribution leads to higher playing continuity or less playing delay. If a user wants to obtain the optimal streaming quality in the future, it should not cheat about its buffer map and needs to honestly forward the available segments.

As explained above, an important design goal of our service-differentiation based incentive algorithms is to defend the potential cheating behavior of overlay nodes during the fulfillment of incentive algorithms. The truthful fulfillment of the service-differentiation algorithm is realized in the contribution-evaluation algorithm we design in the following subsection.

B. Contribution-Evaluation Algorithm

The contribution index of node i is accumulated by its forwarding behaviors. If node i forwards segment s to node k, the contribution of this behavior to $Ct(i)$ is denoted as $Co(i,s,k)$.

1) Computing-Node Selection

For practical consideration, the contribution index of each node is computed distributedly among overlay nodes, and the computing result is sent to the source node. We also want to make the contribution-evaluation algorithm trustworthy itself so that the selfish computing nodes will compute and report the contributions truthfully.

If $Co(i,s,k)$ is computed by node i, node i may cheat about the result when reporting to the source node. Thus, we suggest computing $Co(i,s,k)$ by the segment-requesting node k after node k has received segment s from node i, as illustrated in Fig. 3.

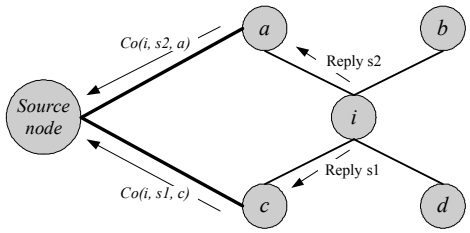

Fig. 3 Contribution evaluation on segment-receiving nodes

Then we prove it is trustworthy to compute $Co(i,s,k)$ in this way. There are two potential cheating behaviors of node k when it reports $Co(i,s,k)$ to the source node: adding the actual value or reducing the actual value.

Lemma 1: node k will not add $Co(i,s,k)$.

Proof: Adding the contribution of node i will not bring any additional benefit to node k, because the benefit of node k already maximizes when it tells the truth. Further, if adding the contribution of node i, when node k and node i are both requesting some segments from another segment-providing node afterwards, node k may suffer some damage since it increases the contribution index of its rival, node i. Therefore, selfish and rational node k will not add $Co(i,s,k)$. ■

Lemma 2: node k will not reduce $Co(i,s,k)$.

Proof: If node k reduces $Co(i,s,k)$, node i may suffer damage in receiving segments from its segment-providing nodes. Since node k selects node i as its providing node for segment s, node i has a great probability to be the "best" providing node to node k for other segments. The damage node i suffers from receiving segments will also have negative impact on node k itself. Therefore, selfish and rational node k will not reduce $Co(i,s,k)$. ■

Both lemma 1 and lemma 2 suggest that the rational choice for node k is to truthfully calculate the contribution of node i and to truthfully report the result to the source node. Therefore, choosing node k to compute $Co(i,s,k)$ after receiving segment s from node i is a reasonable way for both practical and trustworthy consideration.

2) Contribution-Evaluation Algorithm

Then we see how to evaluate *Co(i,s,k)* on node k.

Algorithm 1:

$$Co(i,s,k) = R$$

where R is a constant.

Algorithm 1 is the contribution-evaluation way most commonly used in other studies alike. Since each segment is of uniform length, after receiving segment s from node i, node k evaluates the contribution of node i for this behavior as a constant R. However, it is not so simple for streaming in DONet. Whether segment s arrives at node k no later than the playback time will affect the playing continuity of node k. In addition, the earlier segment s arrives at node k, the playing delay of node k would be less, and there are more chances for node k to forward segment s to other nodes. Therefore, when evaluating $Co(i,s,k)$, the time difference between the segment arrival time and the segment playback time of segment s on node k should be considered. If $Ar(s,k)$ denotes the arrival time of segment s on node k, $De(s,k)$ denotes the playback time of segment s on node k, Tl is the longest tolerance time for a node to wait for a coming segment, we can get a more reasonable evaluation algorithm as algorithm 2.

Algorithm 2:

$$Co(i,s,k) = \begin{cases} 0, \ if \quad Tl < Ar(s,k) - De(s,k) \\ \dfrac{R*[Tl + De(s,k) - Ar(s,k)]}{Tl}, otherwise \end{cases}$$

According to algorithm 2, if segment s arrives at node k earlier than the segment playback time, the contribution of node i will be greater than R; if segment s arrives at node k later than the segment playback time but within the tolerance time, the contribution of node i will be less than R but more than zero; and if segment s arrives at node k beyond the longest tolerance time, it will be dropped and the contribution of node i is zero.

Algorithm 2 seems to be enough. But in order to defend the potential selfish behavior of overlay nodes during the fulfillment of the service-differentiation algorithm designed in the previous subsection, it still needs to be improved. To reduce the computation burden, the segment-providing node might not assign preference indices to the arriving requests as the algorithm demands, but sends the requesting segments in a random order, or in an FIFO way instead. According to the contribution-evaluation algorithm 2, the contribution of the segment-providing node will not decrease if it cheats in this way. To solve this problem, we weight $Pr(k,s)$ in evaluating $Co(i,s,k)$, as algorithm 3.

Algorithm 3:

$$Co(i,s,k) = \begin{cases} 0, if \quad Tl < Ar(s,k) - De(s,k) \\ \dfrac{R*Pr(k,s)*[Tl + De(s,k) - Ar(s,k)]}{Tl}, otherwise \end{cases}$$

In algorithm 3, the higher preference index of the request from node k for segment s is, node i contributes more after forwarding segment s to node k. Under this algorithm, in order to maximize the contribution index of its own, the selfish segment-providing node is motivated to satisfy the requests with higher preference indices earlier. This is in conformity to the service-differentiation algorithm we design.

The contribution-evaluation algorithm is illustrated by Tab. III.

TABLE III. Contribution-evaluation algorithm on node k after receiving segment s from node i

```
1   void contribEvaluate (){
2     if Tl<Ar(s,k)-De(s,k)
3       return
4     else
        // get the contribution index of its own from source node
5       Ct(k)←contReceiveFromSource (k)
6       Pr(k,s) ← Ct(k)/(s+1)   // priority calculation
        // contribution evaluation of node i
7       Co(i,s,k) ← (R*Pr(k,s)*[Tl+De(s,k)-Ar(s,k)])/Tl
8       msg.node←i
9       msg.cont←Co(i,s,k)
        // send the computing result to source node
10      msgSendtoSource(msg)
11      return
12    end if
13  }
```

V. Performance Evaluation

In this section, we analyze the complexity of our incentive algorithms, including both the computation complexity and the communication overhead. In all the following discussions, we suppose the number of nodes in the overlay network is N, the number of partner nodes each node maintains is M, and the number of segments of the propagated stream is Q.

A. Computation Complexity

According to Tab. II, the maximal computation complexity of the service-differentiation algorithm occurs on the node providing all segments to each partner node, which is $O(Q*M*\log M)$, where $\log M$ is the computation complexity of inserting a new-coming request into the requesting queue based on its preference index. The average computation complexity of the algorithm for all nodes is $O(Q*N*\log M/N) = O(Q*\log M)$.

As for the contribution-evaluation algorithm shown in TABLE II, the computation burden is to evaluate the contribution of the segment-providing node each time receiving a segment, and the computation complexity for each node is $O(Q)$.

It is noted that the computation load on each node is unrelated with the network size N. Thus the algorithms are suitable in large-scale overlay networks.

B. Communication Overhead

The communication overhead of our algorithms comes from the communication between each participating node and the source node.

The communication overhead of the service-differentiation algorithm is that each segment-providing node requests the source node for the contribution indices of the segment-requesting nodes, and the source node replies to the segment-providing node. If the communication occurs when each node

receives a segment request from a partner node, the communication overhead of the service-differentiation algorithm in the overlay network is $O(Q*N)$.

Similarly, the communication overhead of the contribution-evaluation algorithm is that each time a segment-requesting node receives a segment, it requests the source node for the contribution index of its own, the source node replies to it, and it reports the contribution of the segment-providing node to the source node. The communication overhead of the contribution-evaluation algorithm is also $O(Q*N)$.

The communication overhead of the overlay network can be reduced by improving the algorithms as follows. In the service-differentiation algorithm, each segment-providing node may request the contribution indices of the partner nodes periodically from the source node, not each time a new request arrives. In the contribution-evaluation algorithm, each segment-requesting node can also request the contribution indices of its own periodically from the source node. And it may not report the contribution of the segment-providing node to the source node each time it receives a segment; instead, the reports can be sent in one single message after receiving several segments.

Based on the analyses above, we can demonstrate that our incentive algorithms are light-weighted and it will not bring too much computing complexity and communication overhead to the DONet.

VI. Conclusion and Future Work

DONet is especially suitable for live stream applications. However, selfish overlay nodes might cheat about their buffer maps to reduce the forwarding burden. To solve this problem, we design a light-weighted service-differentiation based incentive mechanism in this paper, which is composed of the service-differentiation algorithm and the contribution-evaluation algorithm. The primary characteristic of the mechanism is that the contribution evaluation of each node considers the specific characteristic of live stream, and the potential cheating behaviors of overlay nodes during the fulfillment of incentive algorithms are defended.

However, we do not consider the occasion of collusion in this paper. For example, two nodes may drum for the contribution of each other by lying to the source node that they have received a certain segment from each other. We also suppose that each selfish node is rational, meaning that it will not behave against its own interests. If some malicious nodes want to damage the benefits of other nodes in spite of its own interests, the scenario will be more complex. Further techniques need to be introduced to defend collusion and malicious nodes, such as detection and punishment.

Reference

[1] J. Liu, B. Li, and Y. Q. Zhang, "Adaptive video multicast over the Internet," *IEEE Multimedia*, 10(1):22-31, 2003

[2] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs", In *Proceedings of ACM SIGCOMM'88*, Stanford, CA, USA, Aug 1988

[3] Y. H. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast", In *Proceedings of ACM Sigmetrics'00*, Santa Clara, California, USA, Jun 2000

[4] S. Banerjee, B. Bhattacharjee, C. Kommareddy, "Scalable Application Layer Multicast", In *Proceedings of ACM SIGCOMM'02*, Pittsburgh, PA, USA, Aug 2002

[5] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An Efficient Peer-to-Peer Scheme for Media Streaming," In *Proceedings of IEEE INFOCOM'03*, San Franciso, CA, USA, Mar/Apr 2003

[6] Y. Chawathe, "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service", *Ph.D. Thesis*, University of California, Berkekey, Dec 2000

[7] P. Francis, "Yoid: Extending the Internet Multicast Architecture", *White Paper*, http://www.icir.org/yoid

[8] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users", In *Proceeding of IEEE INFOCOM'02*, New York, NY, USA, Jun 2002

[9] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", In *Proceedings of USITS'01*, Boston, MA, USA, Jun 2001

[10] X. Zhang, J. Liu, B. Li, and T. P. Yum, "Data-Driven Overlay Streaming: Design, Implementation, and Experience", In *Proceedings of IEEE INFOCOM'05*, Miami, Florida, USA, Mar 2005

[11] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," In *Proceedings of ACM SIGMETRICS'03*, San Diego, CA, USA, Jun 2003

[12] P. Eugster, R. Guerraoui, A.M. Kermarrec, and L. Massoulie, "From Epidemics to Distributed Computing," *IEEE Computer*, 37(5): 60-67, 2004

[13] S. Jin and A. Bestavros, "Cache-and-Relay Streaming Media Delivery for Asynchronous Clients," In *Proceedings of NGC'02*, Boston, MA, USA, Oct 2002

[14] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous Streaming Multicast," *IEEE Journal on Selected Areas in Communications*, 22(1): 91-106, 2004

[15] J. Shneidman and D. C. Parkes, "Rationality and Self-Interest in Peer to Peer Networks", In *Proceedings of IPTPS'03*, Berkeley, CA, USA, Feb 2003

[16] S. Yuen, and B. Li, "Strategyproof Mechanisms for Dynamic Multicast Tree Formation in Overlay Networks", In *Proceedings of IEEE INFOCOM'05*, Miami, Florida, USA, Mar 2005

[17] W. Wang, X. Li, Z. Suny, and Y. Wang, "Design Multicast Protocols for Non-Cooperative Networks", In *Proceedings of IEEE INFOCOM'05*, Miami, Florida, USA, Mar 2005

[18] P. Golle, K. Leyton-Brown, and I. Mironov, "Incentives for Sharing in Peer-to-Peer networks", In *Proceedings of ACM EC'01*, Tampa, Florida, Oct 2001

[19] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing", In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, California, Jun 2003

[20] C. Buragohain, D. Agrawal, and S. Suri, "A Game Theoretic Framework for Incentives in P2P Systems," In *Proceedings of P2P'03*, Linköping, Sweden, Sep 2003

[21] M. Nowak and K. Sigmund, "Evolution of Indirect Reciprocity by Image Scoring," *Nature*, 393:573–577, 1998

[22] R. Ma, S. Lee, J. Lui, and D. Yau, "A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks," In *Proceedings of ACM SIGMETRICS'04*, New York, NY, USA, Jun 2004

[23] A. Habib and J. Chuang, "Incentive Mechanism for Peer-to-Peer Media Streaming", In *Proceedings of IWQoS'04*, Montreal, Canada, Jun 2004

[24] Y. Cui, D. Li, and J. Wu, "Impact of Buffer Map Cheating on the Streaming Quality in DONet", *Technical Report*

[25] D. Li, Y. Cui, K. Xu, and J. Wu, "Segment-Sending Schedule in Data-Driven Overlay Network", In *Proceedings of IEEE ICC'06*, Istanbul, Turkey, 11-15 Jun, 2006