Defending Against Buffer Map Cheating in DONet-Like P2P Streaming

Dan Li, Jianping Wu, and Yong Cui

Abstract—**Data-driven Overlay Network (DONet)-like P2P system is especially suitable to support live stream applications, since its data structure can tolerate node dynamics quite well. However, optimal streaming demands the cooperation of individual nodes. If selfish nodes cheat about their buffer maps to reduce the forwarding burden, the overall streaming quality would be negatively affected. To defend against this kind of cheating, 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 features of live streaming, not just by the transferring bytes. Secondly, the potential cheating behavior of overlay nodes during the fulfillment of incentive algorithms can be avoided, which is usually not considered by other similar studies. Extensive simulations suggest that the algorithms are indeed effective for defending against buffer map cheating in DONet-like P2P streaming.**

Index Terms—**Buffer map cheating, P2P streaming, trustworthy incentive mechanism.**

I. Introduction

THE rapid growth of Internet has witnessed the wide use of multimedia applications, especially live stream applications [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 multireceiver 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]–[10].

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. Current overlay multicast proposals can be further divided into two types according to the characteristic of data structure, that is, tree-based overlay multicast [3]–[5] and mesh-based overlay multicast [6]–[10]. Like in IP multicast, the data structure in tree-based overlay multicast is also a tree, and data is propagated along the tree after the tree's establishment. However, overlay nodes are unstable. There is 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, bringing negative impact on live stream applications which have stringent demands on streaming continuity.

Meanwhile, 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 protocol belonging to this type [6]. In DONet-like P2P systems, 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 according to certain policies. 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-like P2P systems might cheat about their buffer maps to reduce the forwarding burden to other nodes, which is to be detailed in Section III. In such noncooperative scenarios, some nodes cannot get the true information to request segments, and thus the streaming quality of the whole overlay network might not be optimized.

To defend against buffer map cheating and maximize the streaming quality in selfish DONet-like system, 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 contribution to other nodes. As a reward, nodes with higher contribution indices will obtain better services when they request segments from other nodes. The incentive mechanism is composed

Manuscript received August 28, 2007; revised October 31, 2008. First published March 06, 2009; current version published March 18, 2009. This work was supported in part by the National Natural Science Foundation of China (60873252, 60873253), in part by the National High Technology Development Program of China (2006AA01Z205), in part by the National Major Basic Research Program of China (2009CB320501, 2009CB320503, 2007CB307105), and in part by the International S&T Cooperation Program of China (2008DFA11630). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ishfaq Ahmad.

D. Li is with Microsoft Research Asia, Beijing 100190, China (e-mail: li.dan@microsoft.com).

J. Wu and Y. Cui are with the Department of Computer Science, Tsinghua University, Beijing 100080, China (e-mail: jianping@cernet.edu.cn; cy@csnet1.cs.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TMM.2009.2012941

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

of two algorithms, namely, *service-differentiation algorithm* and *contribution-evaluation algorithm*. Although the idea of maintaining the contribution of each node and using it for service differentiation is not novel, our incentive algorithm has two desirable characteristics compared with other solutions in this area. Firstly, the contribution of each node is evaluated considering the features 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 avoided, which is usually not considered by other similar studies.

We conduct extensive simulations and the simulation results suggest that the incentive algorithms are indeed effective for defending against buffer map cheating as well as encouraging selfish nodes to fulfill them. In this way, truthful streaming can be achieved in selfish DONet-like P2P systems.

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

In tree-based overlay multicast, each node selects a longtime 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], etc.

In mesh-based overlay multicast, there is no explicit parent/children relationship. The data structure used to propagate the stream is a mesh. Therefore, it can tolerate node dynamics well and is especially suitable to support live stream. Representatives of this type of protocols include gossip-based protocols and DONet-like protocols. In gossip-based protocols [7]–[10], each node forwards available data to a set of randomly selected nodes. But in DONet-like protocols, each node maintains several partner nodes, and data is transmitted among partner nodes in a request-reply way, eventually to the whole overlay network. Compared to gossip-based protocols, the advantage of DONet-like protocols is that data is flowing to nodes when they really need it, thus there is no redundant data consuming the precious network bandwidth. Although the controlling packets such as buffer map exchange and data request also consume some resource, the overhead is trivial compared with data stream packets.

An important characteristic of overlay network is that the overlay nodes are all selfish and strategic. The selfish nodes might cheat about their private information to obtain higher interests, which could negatively affect the performance of the whole overlay network. Possible cheating behaviors include distance cheating [22], [23], node throughput cheating [12], relaying cost cheating [13], and etc. In DONet-like P2P systems, selfish nodes can cheat about their buffer map to reduce the forwarding burden, which we call buffer map cheating in this paper.

As a whole, there are two approaches to defending against cheating behaviors of selfish nodes, that is, monetary-payment based mechanism and service-differentiation based mechanism. In the monetary-payment based method, each node pays to the society for resource consumption and gets payment in return for resource contribution [11]–[15]. Since payment-evaluation policy and imaginary currency are demanded in such mechanisms, considerable burden is added to the overlay networks. A further problem is who will fulfill the payment policy, which is often not budget-balanced.

The other approach, service-differentiation based mechanism, is what we adopt to defend against buffer map cheating in this paper. In this method, the historical contributions of all nodes are recorded, and nodes with higher contributions will obtain better services from the society. This idea has also been used to solve similar problems, including both file-sharing applications and streaming applications [16]–[19]. Buragohain *et al.* [16] propose a game theoretic framework to provide incentives in P2P system. In their 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.* [17] 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.* [18] 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, which is not the case in streaming applications, because the streaming continuity is more important than just the data availability.

The most related literature with ours is [19], which designs 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 during the fulfillment the incentive algorithm. There is another paper on buffer management in video transmission [24], which proposes a region-based rate control algorithm for transmitting video in real-time over wireless channels. But our work focuses on transmitting data on the overlay network, and solves the cheating problem of selfish relaying nodes.

Though we are not the first to propose the service-differentiation algorithm to solve cheating problem in P2P systems, what differentiate our algorithms in this paper from all solutions above is that, the contribution-evaluation algorithm considers the specific features of live stream, and the potential cheating behavior of selfish nodes when fulfilling the incentive algorithm itself can also be avoided.

III. Buffer Map Cheating in DONet-Like P2P Systems

The stream propagated in a DONet-like system 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 to the requests, the segment-providing node selects some requests in the requesting queue by certain policy and sends the corresponding segments within its outgoing bandwidth. Each node in the system maintains a *buffer window*, which is used to store a number of continuous segments of

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

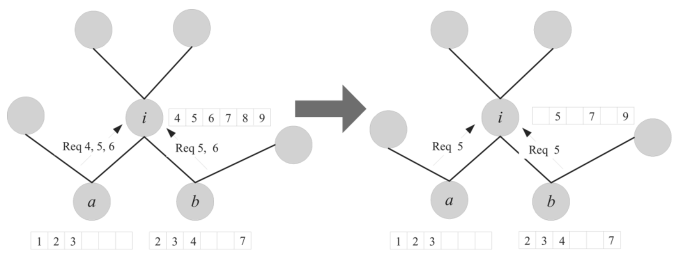

Fig. 1. Buffer map cheating of selfish node i.

the stream. The buffer window size of each node is denoted as B. To make the streaming more continuous, each node does not begin to play the stream immediately after receiving the first segment. Instead, it waits for the arrival of the first several segments and then begins to play the stream. The number of segments each node waits for before playing the stream is defined as the *playing waiting segment number*, denoted by W. Obviously, there is $W \leq B$.

The streaming quality of each node can be evaluated by *playing continuity* and *playing delay*. The playing continuity of node i is defined as the number of segments arriving at node i no later than its playback time over the total number of segments of the stream, denoted as $Pc(i)$. Also the playing delay of node i is defined as the average source-to-end delay of all segments playing on node i. Obviously, higher playing continuity and less playing delay indicate better streaming quality.

Overlay nodes are all selfish and strategic agents that have their own interests. To bear less forwarding burden, selfish nodes might cheat about their private information when exchanging it with partner nodes, so as to reduce the probability of being selected as the segment-providing node by partner nodes. A potential cheating behavior is to cheat about the buffer map information, which is named as *buffer map cheating* in this paper. We first take a glance at it. 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 as *good* as node i.

There may be a lot of selfish nodes cheating like node i in Fig. 1. We can establish the model of buffer map cheating in DONet-like systems as follows. Suppose the total number of nodes in the overlay network is N, the number of partner nodes each node maintains is M, the number of segments of the stream propagated is Q, the buffer window size is B, the playing waiting segment number is W, and the percentage of cheating nodes over all nodes is P. When exchanging the buffer map with partner nodes, each cheating node hides some of its available segments, and the ratio of hidden segments over the total available segments is H. This model of buffer map cheating is denoted as $C(N, M, Q, B, W, P, H)$, where $N > 0$, $M > 0, 0 < W \leq B \leq Q, 0 \leq P \leq 1$, and $0 \leq H \leq 1$.

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, or even cannot find partner nodes to request. As a result, their streaming qualities will be reduced. From the view of the whole system, the buffer map cheating will negatively impact the desired data flow, and the performance of the whole overlay network will not be optimized.

IV. Design of Trustworthy Service-Differentiation-Based Incentive Mechanism

In order to achieve truthful streaming in selfish DONet-like system and thus optimize the whole overlay network, we design a service-differentiation based incentive mechanism to defend against buffer map cheating described in the section above. Our design goal is that each selfish and rational node will be guided to advertise the actual buffer map information under the incentive mechanism, and that the potential cheating behavior of selfish nodes during the fulfillment of the incentive mechanism itself can also be avoided.

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 to 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 stream session. The source node can play as the role of a trustworthy third-party, because its benefit is associated with the overall outcome of the stream 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: *service-differentiation algorithm* and *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 with 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 with its preference index, and accordingly put into the requesting queue of the segment-providing node. When the segment-providing node replies to 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 with 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 nondecreasing over the contribution index. An example of mapping function is shown in the following:

$$Pr(i, s) = Ct(i). \tag{1}$$

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

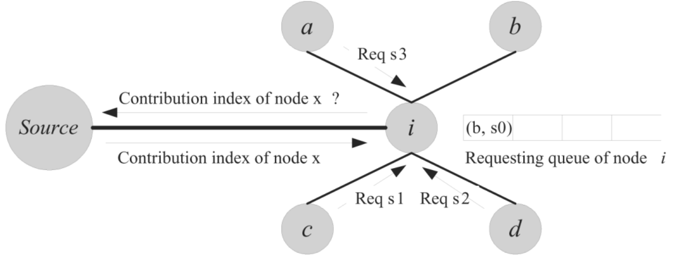
a
b
Req s3
Contribution index of node x ?
Source
i
(b, s0)
Contribution index of node x
Requesting queue of node i
Req s1
Req s2
c
d

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 obviously not in favor of stream propagation among 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. According to [20], lower-sequence favored streaming will reduce the playing delay. Therefore, when assigning the preference indices to the segment requests, the requests for lower-sequence segments are also favored.

After considering the two factors, the preference index of a request is nondecreasing over the contribution index of the requesting node, and is nonincreasing over the sequence number of the requesting segment. The example mapping function shown in (1) can be extended to the following:

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

The service-differentiation algorithm on the segment-providing node can be illustrated in Table I.

After introducing the service-differentiation algorithm, 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 node hopes obtaining the optimal streaming quality in future, it should not cheat about its buffer map and needs to honestly forward the available segments.

As previously stated, an important design goal of our service-differentiation based incentive mechanism is to avoid the potential cheating behavior of overlay nodes during the fulfillment of incentive mechanism itself. The truthful fulfillment of the service-differentiation algorithm is achieved 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 $Co(i,s,k)$. We thus primarily discuss how to compute $Co(i,s,k)$.

1) Computing-Node Selection: For practical consideration, the contribution index of each node is computed distributively 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.

TABLE I
SERVICE-DIFFERENTIATION ALGORITHM ON THE SEGMENT-PROVIDING NODE

```
// processing an arriving segment request
void reqMsgProcess($reqMsg$){
    $k \leftarrow reqMsg.source$
    $s \leftarrow reqMsg.segment$
    // obtaining the contribution index of node $k$ from the source
    $Ct(k) \leftarrow$ contReceiveFromSource($k$)
    //preference index assignment
    $Pr(k,s) \leftarrow \frac{Ct(k)}{s+1}$
    //insert the request into the requesting queue according to its
    // preference index
    queueInPr($k, s, Pr(k,s)$)
    **return**
}
```

```
//service differentiation for requests in the requesting queue
void serviceDiff(){
    **while** (**true**)
        // there is request in the requesting queue
        **if** !queueEmpty()
            // get the requests near the queue head within the
            // outgoing bandwidth
            queueOut (&*reqs*)
            // reply the requests by sending the requesting
            // segments to the corresponding requesting nodes
            segmentSend(*reqs*)
        **end if**
    **end while**
    **return**
}
```

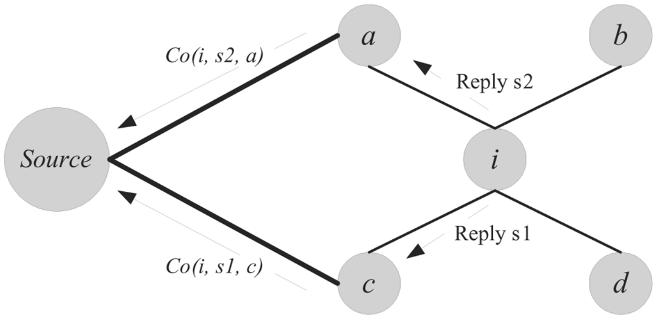

Fig. 3. Contribution evaluation on segment-receiving nodes.

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.

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, that is, adding the actual value or reducing the actual value.

Lemma 1: Any rational node k has no incentive to add the contribution value of node i for forwarding segment s, $Co(i,s,k)$.

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 $Co(i,s,k)$.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

Lemma 2: Any rational node k has no incentive to reduce the contribution value of the node i for forwarding segment $s, Co(i,s,k)$.

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 bring 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 discuss 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 method 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-like systems. 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 (except that the playing waiting segment number is only 1, which is not the common case), 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,[1] we can get a more reasonable evaluation algorithm as algorithm 2.

Algorithm 2: See the first equation shown at the bottom of the page. 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.

[1]Indicating that the node would wait for the segment for an additional time of Tl even after its playback deadline, and the stream is paused during such an interval; but there is no such tolerance time if we assume $Tl = 0$.

TABLE II
CONTRIBUTION-EVALUATION ALGORITHM ON NODE k AFTER RECEIVING SEGMENT s FROM NODE i

```
// processing an arriving segment request
void contribEvaluate (){
    if $Tl < Ar(s,k) - De(s,k)$
        return
    else
        // get the contribution index of its own from source
        $Ct(k) \leftarrow$ contReceiveFromSource (k)
        // priority assignment
        $Pr(k,s) \leftarrow \frac{Ct(k)}{s+1}$
        // contribution evaluation of node $i$
        $Co(i,s,k) \leftarrow \frac{R*Pr(k,s)*[Tl+De(s,k)-Ar(s,k)]}{Tl}$
        $msg.node \leftarrow i$
        $msg.cont \leftarrow Co(i,s,k)$
        // send the computing result to source
        msgSendtoSource(msg)
        return
    end if
}
```

Algorithm 2 seems to be enough. But in order to avoid the potential cheating behavior of overlay nodes during the fulfillment of the service-differentiation algorithm designed in the previous subsection, it needs further improvement. 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 at all if it cheats in this way. To avoid this problem, we weight $Pr(k,s)$ in evaluating $Co(i,s,k)$, as algorithm 3.

Algorithm 3: See the second equation shown at the bottom of the page.

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 respond to the requests with higher preference indices earlier. This is in consistent with the service-differentiation algorithm we design.

The contribution-evaluation algorithm is illustrated by Table II.

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

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

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

C. Algorithm Complexity

In this subsection, we analyze the complexity of our incentive algorithms, including both the computation complexity and the communication overhead. As in the model of buffer map cheating, 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.

1) Computation Complexity: According to Table 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 $O(Q * N * \log M/N) = O(Q * logM)$.

As for the contribution-evaluation algorithm shown in Table I, 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.

2) 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 whole 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 in turn, and finally it reports the contribution of the segment-providing node to the source node. The communication overhead of the contribution-evaluation algorithm in the whole overlay network 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. Also 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.

V. Simulations

Though there are many service-differentiation based proposals against node selfishness in P2P networks, as we discussed in Section II, our solution in this paper is the only one that considers the selfishness problem during the fulfillment of the incentive algorithm itself. Hence, we only conduct extensive simulations to study the effectiveness of our solution, using the cheating model $C(N, M, Q, B, W, P, H)$ established in Section III.

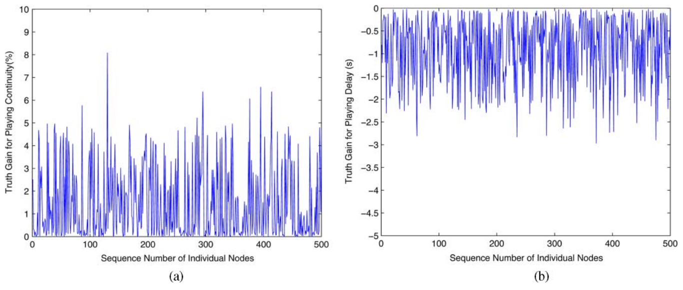

Fig. 4. Truth gain when $B = 70, W = 10, H = 50\%$. (a) Playing continuity. (b) Playing delay.

We use the random model to generate the network-layer topology by GT-ITM toolkit [21]. There are 2000 routers in the network-layer topology and the link distances between connected routers are within [10 ms, 500 ms]. Suppose the number of overlay nodes participating in the system is 500 $(N = 500)$, and they are attached to different routers randomly selected among the 2000 routers. According to [6], the number of partner nodes each node maintains is recommended as 4 $(M = 4)$. The stream propagated in the overlay network is composed of 5000 segments $(Q = 5000)$.

Firstly, we study the *truth gain* of each node (Sim. 1 and 2), which indicates whether a node has the incentive to fulfill our algorithm. The *truth gain* of a node is evaluated as the streaming quality (playing continuity or playing delay) when telling the truth minus that when cheating, given all other nodes tell the truth.

Secondly, we study the *system improvement* of the whole overlay network (Sim. 3, 4, and 5), which implies whether the system will indeed benefit from our incentive algorithm. Under our algorithm, no rational nodes will cheat about their available segments, i.e., the cheating node percentage is 0%. We compare the average streaming quality of all nodes when the cheating node percentage (P) is 0% with that of 10%, 30%, 50%, 70%, and 90%.

A. Truth Gain

The truth gain of each node is measured on the two metrics of streaming quality respectively, that is, playing continuity (Pc) and playing delay (Pd). The cheating degree in the cheating model is set as 50% $(\mathrm{H} = 50\%)$, indicating that each node hides half of its available segments. To investigate the truth gain with diverse system configurations, we conduct several groups of simulations, and two of them are reported here (Sim. 1 and 2). In fact, the results of the other groups also support the same conclusion. In Sim. 1, we set the buffer window size (B) as 70, and the playing waiting segment number (W) as 10. In Sim. 2, we let $B = 90$ and $W = 30$.

Simulation 1: The truth gains of each node for playing continuity and playing delay under the setting of Sim. 1 are shown in Fig. 4(a) and (b), respectively.

Fig. 4(a) tells that the truth gains of all nodes for playing continuity are nonnegative, meaning that advertising the actual buffer map would bring higher playing continuity than cheating. From Fig. 4(b), we find that the truth gains of all nodes for playing delay are all nonpositive, suggesting that telling the truth would reduce the playing delay.

Simulation 2: The truth gains of each node for playing continuity and playing delay under the setting of Sim. 2 are shown in Fig. 5(a) and (b), respectively.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

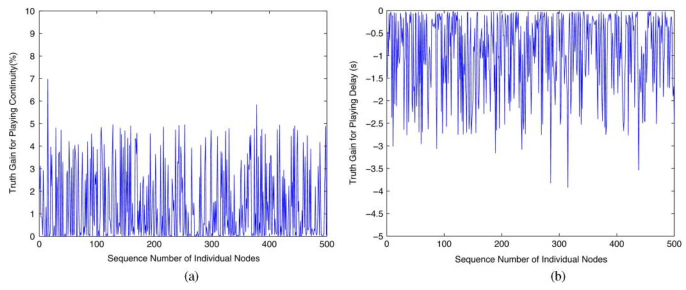

Fig. 5. Truth gain when $B = 90, W = 30, H = 50\%$. (a) Playing continuity. (b) Playing delay.

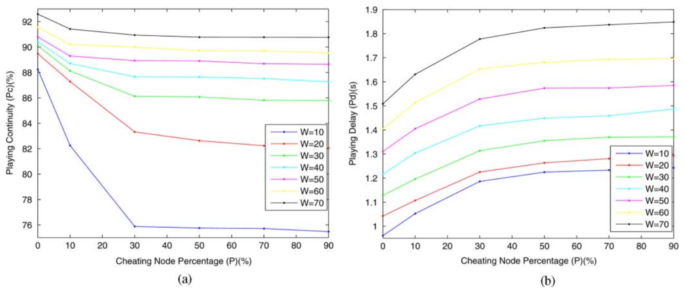

Fig. 7. System improvement when $B = 70, H = 50$. (a) Playing continuity. (b) Playing delay.

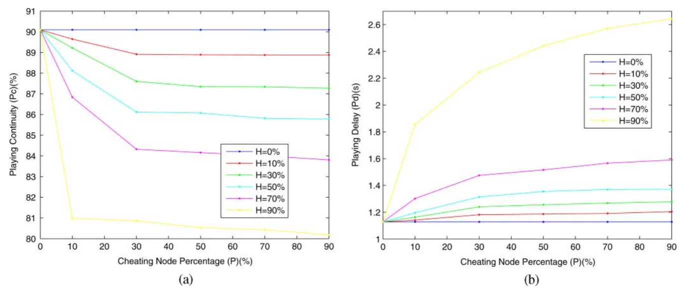

Fig. 6. System Improvement when $B = 70, W = 30$. (a) Playing continuity. (b) Playing delay.

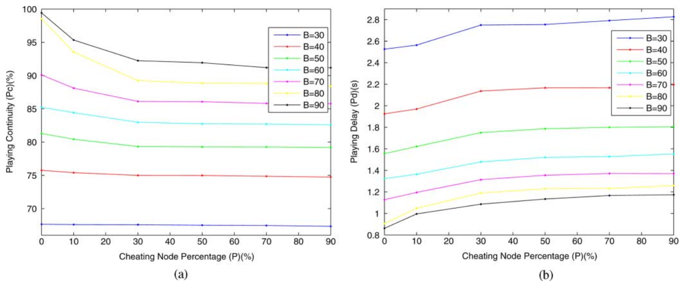

Fig. 8. System improvement when $W = 30, H = 50$. (a) Playing continuity. (b) Playing delay.

Similar with Fig. 4(a), Fig. 5(a) shows that the truth gains of all nodes for playing continuity are nonnegative. Also Fig. 5(b) illustrates that the truth gains of all nodes for playing delay are nonpositive, in consistence with Fig. 4(b).

From all the simulation results above of truth gain, we see that the values of truth gains are considerable to incentivize selfish nodes to tell the truth. Also we should note that the values are controlled because we consider the sequence number of the requested segment when assigning the preference index for segment requests so as to avoid starvation. If not, the values of truth gains would be even larger.

B. *System Improvement*

To study the system improvement after introducing the incentive algorithm, we compare the average playing continuity and playing delay of all nodes when the cheating node percentage (P) is 0% (which means that there is no node cheating) with that of 10%, 30%, 50%, 70%, and 90%. Similarly, we conduct three experiments to investigate the diverse system configurations, namely, Sim. 3, 4, and 5. In Sim. 3, we let the buffer window size $B = 70$, the playing waiting segment number $W = 30$, and the cheating degree H vary as 0%, 10%, 30%, 50%, 70%, and 90%. In Sim. 4, we let $B = 70$, $H = 50\%$, and W vary as 10, 20, 30, 40, 50, 60, and 70. In Sim. 5, we let $W = 30, H = 50\%$, and B vary as 30, 40, 50, 60, 70, 80, and 90.

Simulation 3: The playing continuity and the playing delay under the setting of Sim. 3 with different cheating node percentages are shown in Fig. 6(a) and (b), respectively.

From Fig. 6(a), we see that under all cheating degrees higher than 0%, the maximal playing continuity is achieved when there is no node cheating. When the cheating node percentage increases, the playing continuity decreases, but the decreasing gets smoother. We also find that the playing continuity is higher with lower cheating degree. When the cheating degree is 0%, which means that there is no node cheating, the playing continuity gets the optimal value. The change of playing continuity is especially steep when the cheating node percentage is low and the cheating degree is high.

Fig. 6(b) shows that under all cheating degrees higher than 0%, the playing delay is least when there is no node cheating. As the cheating node percentage grows, the playing delay becomes more. In addition, the lower is the cheating degree, the more is the playing delay. It is similar to Fig. 4 that the change of playing delay is especially obvious when the cheating node percentage is low and the cheating degree is high.

Simulation 4: The playing continuity under the setting of Sim. 4 with different cheating node percentages is shown in Fig. 7(a), and the playing delay is shown in Fig. 7(b).

Fig. 7(a) illustrates that no matter what the playing waiting segment number is, the playing continuity is maximal when there is no node cheating. The playing continuity decreases with the growth of cheating node percentage, but the curve becomes smoother. Additionally, more playing waiting segment number brings higher playing continuity.

From Fig. 7(b), we conclude that the playing delay increases when the cheating node percentage grows, and is shortest when there is no node cheating, no matter the playing waiting segment number. In addition, when the playing waiting segment number is bigger, the playing delay is longer.

Simulation 5: With different cheating node percentages, the playing continuity and the playing delay under the setting of Sim. 5 are shown in Fig. 8(a) and (b), respectively.

Fig. 8(a) suggests that with all buffer window sizes, the playing continuity is highest when there is no node cheating, and gets lower with more cheating nodes. This is consistent with Figs. 6(a) and 7(a). In addition, given the same cheating node percentage, the bigger buffer window size brings higher playing continuity. The change of playing continuity is most obvious when the buffer window size is big and the cheating node percentage is small.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.

Fig. 8(b) tells that the playing delay increases with the growth of cheating node percentage, and decreases with the growth of buffer window size, telling the similar conclusion as Figs. 6(b) and 7(b).

From all the simulation results above of system improvement, we find that the streaming quality is always optimized when there is no node cheating. Therefore, our service-differentiation based incentive algorithms are indeed effective for defending against buffer map cheating and thus improving the streaming quality in selfish DONet-like systems.

VI. Conclusion

DONet-like P2P systems are especially suitable for live stream applications. However, selfish overlay nodes might cheat about their buffer maps to reduce the forwarding burden. To defend against this kind of cheating, we design a service-differentiation based incentive mechanism in this paper, which includes 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 features of live stream, and the potential cheating behaviors of overlay nodes during the fulfillment of incentive algorithms can be avoided. The performance analyses show the overhead of the algorithms is not high and extensive simulations tell that the algorithms are indeed effective to defend against buffer map cheating as well as encourage selfish nodes to fulfill them. Therefore, truthful streaming can be achieved in selfish DONet-like systems under the service-differentiation based incentive mechanism we design.

References

[1] J. Liu, B. Li, and Y. Q. Zhang, "Adaptive video multicast over the internet," *IEEE Multimedia*, vol. 10, no. 1, pp. 22–31, Jan./Feb. 2003.

[2] S. E. Deering, "Multicast routing in internetworks and extended LANs," in *Proc. ACM SIGCOMM'88*, Stanford, CA, Aug. 1988.

[3] Y. H. Chu, S. G. Rao, and H. Zhang, "Case for end system multicast," in *Proc. ACM Sigmetrics'00*, Santa Clara, CA, Jun. 2000.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM'02*, Pittsburgh, PA, Aug. 2002.

[5] D. A. Tran, K. A. Hua, and T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," in *Proc. IEEE INFOCOM'03*, San Francisco, CA, Mar./Apr. 2003.

[6] X. Zhang, J. Liu, B. Li, and T. P. Yum, "Data-driven overlay streaming: Design, implementation, and experience," in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[7] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proc. ACM SIGMETRICS'03*, San Diego, CA, Jun. 2003.

[8] P. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, May 2004.

[9] S. Jin and A. Bestavros, "Cache-and-relay streaming media delivery for asynchronous clients," in *Proc. NGC'02*, Boston, MA, Oct. 2002.

[10] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast," *IEEE J. Select. Areas Commun.*, vol. 22, no. 1, pp. 91–106, Jan. 2004.

[11] J. Shneidman and D. C. Parkes, "Rationality and self-interest in peer to peer networks," in *Proc. IPTPS'03*, Berkeley, CA, Feb. 2003.

[12] S. Yuen and B. Li, "Strategyproof mechanisms for dynamic multicast tree formation in overlay networks," in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[13] W. Wang, X. Li, Z. Suny, and Y. Wang, "Design multicast protocols for non-cooperative networks," in *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005.

[14] P. Golle, K. Leyton-Brown, and I. Mironov, "Incentives for sharing in peer-to-peer networks," in *Proc. ACM EC'01*, Tampa, FL, Oct. 2001.

[15] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A secure economic framework for P2P resource sharing," in *Proc. Workshop on Economics of Peer-to-Peer Systems 2003*, Berkeley, CA, Jun. 2003.

[16] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in P2P systems," in *Proc. P2P'03*, Linkoping, Sweden, Sep. 2003.

[17] M. Nowak and K. Sigmund, "Evolution of indirect reciprocity by image scoring," *Nature*, vol. 393, pp. 573–577, 1998.

[18] R. Ma, S. Lee, J. Lui, and D. Yau, "A game theoretic approach to provide incentive and service differentiation in P2P networks," in *Proc. ACM SIGMETRICS'04*, New York, Jun. 2004.

[19] A. Habib and J. Chuang, "Incentive mechanism for peer-to-peer media streaming," in *Proc. IWQOS'04*, Montreal, QC, Canada, Jun. 2004.

[20] D. Li, Y. Cui, K. Xu, and J. Wu, "Segment-sending schedule in data-driven overlay network," in *Proc. IEEE ICC'06*, Istanbul, Turkey, Jun. 11–15, 2006.

[21] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996.

[22] L. Mathy and N. Blundell, "Impact of simple cheating in application-level multicast," in *Proc. IEEE INFOCOM'04*, Hong Kong, China, Mar. 2004.

[23] D. Li, Y. Cui, K. Xu, and J. Wu, "Impact of receiver cheating on the stability of ALM tree," in *Proc. IEEE GLOBECOM'05*, St. Louis, MO, Nov./Dec. 2005.

[24] Y. Sun, I. Ahmad, D. Li, and Y. Zhang, "Region-based rate control and bit allocation for wireless video transmission," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 1–10, Jan. 2006.

Dan Li received the M.E. and Ph.D. degrees from Tsinghua University, Beijing, China.

He is currently an Associate Researcher with Microsoft Research Asia, Beijing. His research interests include P2P multimedia communication, data center networks, and social networks.

Jianping Wu received the M.E. and Ph.D. degrees from Tsinghua University, Beijing, China.

He is currently a full Professor in the Department of Computer Science, Tsinghua University. In the research areas of the network architecture, high performance routing and switching, protocol testing, and formal methods, he has published more than 200 technical papers in academic journals and proceedings of international conferences.

Yong Cui received the B.S., M.E., and Ph.D. degrees from Tsinghua University, Beijing, China.

He is an Associate Professor with Tsinghua University, and a Council Member in the China Communication Standards Association. He directed several national R&D projects, funded by Chinese Next Generation Internet Plan (CNGI), National Natural Science Foundation of China (NSFC), and Development Plan of the State High Technology Research of China (863). Having published more than 60 papers in international journals and conferences, he also holds more than 20 patents. His major research interests include wireless mobile Internet, computer network architecture, distributed routing protocols, and P2P networks.

Authorized licensed use limited to: Tsinghua University. Downloaded on June 16,2022 at 04:43:10 UTC from IEEE Xplore. Restrictions apply.