A multi-sender multicast algorithm for media streaming on peer-to-peer networks

Mohammad Hamed Firooz a,*, Keivan Ronasi a, Mohammad Reza Pakravan b, Alireza Nasiri Avanaki a

a Electrical and Computer Engineering University of Tehran, Tehran, Iran
b Sharif University of Technology, Tehran, Iran

Received 4 October 2006; received in revised form 3 May 2007; accepted 11 May 2007
Available online 29 May 2007

Abstract

Unlike those on Internet, the media providers on P2P networks are ordinary nodes with limited shared resources such as bandwidth. Multi-sender methods are the best existing solutions to video streaming on P2P networks. In this paper, we propose use of a multicast method on the top of an arbitrary multi-sender method so that all requesting peers receive almost the same expected bit-rate. Experimental results, derived from implementation of the proposed algorithm on Pastry P2P network confirm our claim. Another advantage of our method over the existing methods is its scalability with the number of receivers.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Peer-to-peer networks; Multi-sender; Overlay multicast; Multimedia streaming; Pastry environment

1. Introduction

Increasing penetration of high-speed Internet access (e.g., ADSL) among the users of peer-to-peer (P2P) networks enables deployment of real-time multimedia delivery schemes over them, in addition to file sharing – their traditional application [1,2]. The main difference of real-time media streaming over P2P and the traditional streaming over IP is that the senders in P2P networks are ordinary nodes with limited bandwidth and availability.

An example solution to media streaming over P2P is GnuStream [3,4] which uses multiple senders to stream a video to the receiver. GnuStream is, however, not robust to the changes in the network topology caused, for example, by some nodes leaving the network. This problem is solved in [5] by introduction of a central power peer responsible for sender selection and switching when one goes offline. In many real-life scenarios, the assumption of an always-available central power node cannot be justified.

Some proposed methods require the knowledge of the (properties of) network topology (e.g., link bandwidths or physical proximities of the neighbors of each node) [6,7]. In [6], the senders that are geometrically closer to the receiver are chosen. This approach decreases the initial streaming delay. However, by ignoring the availability of the nodes in sender selection, the quality of the received video may be sub-optimal.

PROMISE [8,9] is one of the first algorithms tackling the multi-sender problem considering the network topology such as links bandwidths and their error rate as well as the peer characteristics such as their availability. PROMISE finds the set of the senders that maximize the reliable bit-rate offered to the receiver. The main problem with PROMISE is scalability: its computational complexity increases exponentially with the number of senders. This problem is solved in FastIPROMISE [10] which offers a better performance at a dramatically reduced computational cost.
A major problem with all existing multi-sender algorithms is that they provide a solution for only one peer requesting the media at a time. That is, they try to provide the maximum expected bit-rate (EBR) to the first requesting node and when another peer requests the same media, the algorithm should be run again, but the expected bit-rate (EBR) provided to the second requesting node is most likely considerably lower (under the acceptable threshold). A natural solution to this problem is a multicast scheme providing media to several receivers without imposing hard constraints on the senders or the network.

Many application level multicast protocols are proposed [11]. In [12], a method for media streaming to multiple receivers considering some P2P networks characteristics such as limited bandwidths of peers is proposed. The method requires a power sender which may not exist in many cases. Gridmedia [13], another example of application-level multicast protocols, assumes equal upload bandwidths for all nodes.

Itaya and others exploit a combination of multi-sender and multicast to stream media from multiple senders to a number of receivers [14–16]. Their method is not practical because of the following (i) the assumption of unlimited bandwidths of all senders; (ii) the method is not scalable in the sense that the number of control packets communicated among the senders increases exponentially with the number of senders and receivers; and finally (iii) the method does not consider senders’ limited availability.

In this paper, we propose a multi-sender multicast (MSMC) algorithm that maximizes the received bit-rate (almost near average EBR of the first receiver) for all requesting peers, thus overcomes all problems mentioned above. In contrast to [17], the proposed method is receiver-driven. That is, the receiver is in charge of media splitting and streaming control and knows which part is coming from which sender. Our contributions are summarized below.

1. By employing a multi-sender method, the limited bandwidths of the sender peers do not impose a serious restriction on streaming quality. In fact, MSMC can be integrated into any existing multi-sender scheme to provide a scalable multicast solution. Also, by considering the availability of senders, the quality of the streamed media is improved.

2. Another advantage of being receiver-driven is that the multicast trees are made in a distributed manner. Each receiver makes its multicast tree itself. Also in the proposed method, the joining operation is managed by previously joined receivers except for the first receiver which is managed by the senders.

3. By being a multicast scheme, a large number of receivers can receive multimedia from a limited number of senders without stressing the P2P substrate or the senders.

4. Using Pastry [18,19] leads to a simple algorithm in making multicast trees and easy distributed management of those. Neither the receivers nor the senders are involved in multicast tree management.

The only limitation of our proposed method is that, similar to most of the existing P2P multicast schemes [11], we also require cooperation of some uninterested peers (playing application-layer routers) in distributed management and control of streaming. To be fair, such nodes are randomly selected. Therefore, our method must be implemented in a cooperative network in which nodes help each other to get services.

MSMC can be deployed on any P2P substrate which maintains peers connectivity, manages peer membership and performs object lookup. We use FreePastry [18,19] because its code is well documented and easily portable, as well as its simple network management.

The rest of the paper is organized as follows. In Section 2, the proposed method is described. Section 3 explains implementation of MSMC in FreePastry environment. The multi description coding and the buffering system we used are described in Section 4. Section 5 reports our simulation results. The paper is concluded in Section 6.

2. Proposed method

If a receiver R requests a certain multimedia, a set of candidate senders (determined by a location protocol [20]) having the desired media, signal their readiness to transmit data to R. The receiver can simply connect directly to the senders and start downloading. However, as the download bandwidth of a typical node is considerably larger than its upload bandwidth (e.g., for ADSL the ratio is 8 to 1), the simple scheme of direct connection leads to selfish usage of the network resources: no other nodes can use the senders from which R is receiving the media. Also, R cannot provide the media to any other receiver at the same bit-rate it is receiving the media.

In fact, after a limited number of receivers, all senders will be busy and no other receiver can receive the media. Besides, as R selfishly tries to select the best set of senders, the other receiver (joining later) will have the worst quality of service.

To overcome the above mentioned problems, R can connect to the senders via a few application-level virtual routers (a.k.a. forwarders [11]; Fig. 1). Determination of this topology is explained in Section 3.

A multi-sender method (such as IPROMISE [21,10]) selects a subset S of all candidate senders (referred to as active senders), each offering R a bit-rate of ri that minimizes the following.

\[
E \left[ \sum_{i \in S} D_i P(D_i) \right]
\]

subject to

\[
\sum_{i \in S} r_i \geq R_r
\]

where \(D_i\) is the amount of distortion caused by node \(i \in S\) going offline or decreasing its bandwidth, and \(P(D_i)\) is the...
availability, the lower is $P(D_i)$. $R_r$ indicates the minimum bit-rate required at the receiver.

To each node, $P_i$, two parameters are associated: $(A_i, R_i)$. $A_i$ is the availability probability of $P_i$, an indication of how long $P_i$ is online in the unit time interval. $R_i$ is the amount of upload bandwidth $P_i$ shares with the network.

For simplicity, we assume that the bandwidth offered by each peer does not change while it is on the network. Thus, distortion happens if and only if an active sender leaves the network. Eq. (1) can be written as follows.

$$ E \left[ \sum_{i \in S} r_i (1 - A_i) \right] $$

which is simplified to:

$$ E \left[ \sum_{i \in S} r_i - \sum_{i \in S} r_i \cdot A_i \right] = R_r - E \left[ \sum_{i \in S} r_i \cdot A_i \right] $$

in which, the second term gives the bit-rate provided to the receiver (by active senders). $R_r$, as mentioned before is the receiver’s intended play back rate and is fixed. Thus, minimization of (3) is equivalent to maximization of $E[\sum_{i \in S} r_i \cdot A_i]$, the expected bit-rate offered to the receiver.

Our proposed method (MSMC) assumes that a multi-sender algorithm has already determined the subset $S$ offering the maximum EBR for the first receiver $R$ (i.e., the multi-sender algorithm minimized (1) subject to (2)), and for each node $P_i$ in $S$, $r_i$ is the bit-rate transmitted to the receiver. As mentioned before, if the same set of senders are used to serve a second receiver $R'$, almost certainly the EBR provided to $R'$ will be unacceptably low as almost all of the limited upload bandwidths of the senders are already committed to streaming of media to $R$ (Section 5). Either that, or $R'$ has to resort to less qualified senders, if any. Such scheme is therefore not scalable with the number of receivers and cannot be used for streaming of popular media (e.g., soccer match broadcast).

The proposed multicast method is illustrated in Fig. 2. Nodes $S_1$ to $S_4$ are active senders (i.e., have the media and are transmitting data to a receiver) to $R$. Nodes $T_1$ to $T_3$, which are routing the stream in this topology, have a partial content of the streamed media. Our idea is to use these routers as temporary senders to forward copies of packets destined for $R_1$ to another requesting node such as $R_2$. Intuitively, one can observe that using this approach the EBR of $R_2$ can be increased without pressuring the bandwidths of the primary senders. Moreover, by addition of each receiver to this topology, a number of new temporary senders appear that can serve even more new receivers. Thus, the proposed method is scalable with the number of receivers.

After the receiver selects the senders, $S$, it should notify $P_i$, $i \in S$, to begin transmission. The receiver is responsible for informing each sender which part of the media it should transmit. Synchronization and media buffering control are also receiver responsibilities (as mentioned before our algorithm is receiver-driven). First of all, receiver informs each sender about a number $n$ defined as

$$ n = \sum_{i \in S} \text{round} \left( \frac{r_i}{r_{min}} \right) $$

where $r_{min} = \min_{i \in S} (r_i)$.

As defined in (5), $n$ can be considered as the number of virtual senders with bit-rates $r_{min}$. That is, an actual sender with bit-rate $r_i$ is $r_i/r_{min}$ virtual senders each with $r_{min}$ bit-rate grouped together. All actual active senders are informed of the number $n$ and they divide the media stream into $n$ blocks, a task which can performed using multiple

![Fig. 1. A sample multi-sender structure. Virtual routers (a.k.a. “forwarders” [11]) route data to the receiver over the P2P substrate.](image1)

![Fig. 2. Virtual routers can be used as temporary senders to provide the stream to the new receiver.](image2)
description coding (MDC; briefly described in Section 4) [22,23] or multiple layered coding (MLC) [6]. Followed by \( n \), \( \{r_{\text{min}}, j\} \) are also given to each sender, in which \( j \) denotes the number of the first virtual sender in each actual sender. That is, \( P_i \), the \( i \)th actual sender, must transmit data on behalf of the virtual sender \( j \) to \( k + \text{round}(r_i/r_{\text{min}}) - 1 \). In other words, \( P_i \) sends media blocks with indices \( h_i = nd + J \) where

\[
J = (j,j + 1, \ldots, j + \text{round}(r_i/r_{\text{min}}) - 1) \\
\text{and } d = 0, 1, 2, 3, \ldots
\]

For example, suppose senders \( P_1 \) to \( P_4 \) transmit data to the receiver with bit-rates 100, 200, 500, and 200 kbps. The media is divided among 10 virtual senders. Blocks 0, 10, 20, \ldots are sent by \( P_1 \). For \( P_4 \), \( j = 8 \), and it sends blocks with indices 8, 9, 18, 19, 28, 29, \ldots

The \( i \)th section is defined as the \( i \)th sender and the routers on its path to the receiver. Media packets with indices \( nd + J \) are available in the \( i \)th section. If \( R_{ij} \) is bit-rate of node \( j \)th over \( i \)th section, the number \( n' \) is defined as follows.

\[
n' = \sum_j \text{round} \left( \frac{R_{ij}}{\text{min}(R_{ij}) = R_{\text{min}}} \right)
\]

Analogous to sender \( P_i \), node \( P_{ij} \) (although it is a temporary sender, we call it a sender) is equivalent to \( R_{ij}/R_{\text{min}} \) virtual senders with \( R_{\text{min}} \) as their bit-rate. The second receiver, \( R' \), should send to each of the nodes (\( P_i \) and all temporary senders) \( \{n', k, R_{\text{min}}\} \) in which \( k \) denotes the first virtual sender on that section. Therefore, \( P_{ij} \) should forward packets with indices \( h_{ij} = n'd + K_{ij} \) among all packets on the \( i \)th section. In fact, from \( R' \) perspective, each virtual router is a sender which should be given \( \{r_{\text{min}}, j\} \).

If \( \sum R_{ij} < R_i \), then \( P_i \) and all temporary senders on its section cannot forward all packets on this section to \( R' \). \( R_{\text{hid}} = R_i - \sum R_{ij} \) is defined as the amount of bit-rate not forwarded to the second receiver. In this case, a multi-sender algorithm should be run to provide \( \sum R_{\text{hid}} \) bit-rate for \( R' \).

Similar to other multicast applications, our method is developed for live video streaming: if a receiver joins the network 5 min late, it can either enjoy the rest of the broadcast with the rest of the receivers at high quality, or start watching the media from the beginning at a lower quality by running the multi-sender algorithm alone (i.e., without benefiting from the multicast scheme).

3. Implementation of the proposed method in Pastry

3.1. An introduction to Pastry

Pastry is a scalable, decentralized, fault-tolerant, self-organizing, structured network used in very large networks overlaid on Internet. Content location. Each node has a 128-bit ID which is the value of a hash function for its IP address. Each content has an ID which can be, for example, the value of the hash function for its descriptive text. A pointer to content is placed in the node with the closest ID to the content ID. 128-bit IDs are divided into \( b \)-bit digits ranging from 0 to \( B - 1 \): \( B = 2^b \) (\( b \) is typically 4).

In order to find a particular content having its ID, one should find the node with the closest ID to the target ID. For this purpose, each Pastry node \( (X) \) keeps three tables (Fig. 3). (1) Routing table \( (R) \) with \( \log_B N \) rows and \( B - 1 \) columns. The entry at row \( i \) and column \( j \) is the IP address of the closest available node whose ID’s first \( i - 1 \) digits is equal to that of node \( X \) and the \( i \)th digit is \( j \) not equal to that of the present node. If such node is not found, the corresponding entry is left empty. (2) Leaf set \( (L) \): it has \( L \) entries in which the IP addresses of \( L \) numerically nearest available IDs \((L/2 \text{ smaller} + L/2 \text{ larger than the present node})\) are contained. Typically, \( L = B \). (3) Neighborhood set \( (M) \): it contains the IP addresses of \( M \) physically nearest nodes to the present node. This table is not used for routing and is used to maintain locality. Typically, \( M = 2B \).

We briefly explain content location using the routing table (see [19] for details). To find a specific content, given its ID the node checks if the ID of the desired content lies within its leaf table. If so, the location request is passed to that node in the leaf set with the closest ID to the desired content. If not, the node considers in how many digits its ID is the same as the target node’s. Then it searches the considered row of its routing table, trying to make the next digit equal to the target by browsing the appropriate column. The request is sent to the entry found in this way. If this entry was empty, it finds the nearest node by

![Fig. 3. Pastry table for a conceptual node: 4-byte IDs are used for simplicity.](image-url)
searching all its three tables and sends the location request to it. The location process is continued by the new node who received the location request.

In this way, a request travels through the Pastry nodes, getting closer to the target in each step. That is because the number of common digits increases in each step until the desired content is reached.

The pseudo code of the above mentioned algorithm is given in Fig. 4 when key \( \text{D} \) arrives at a node with node ID \( \text{A} \). Following notations are used:

- \( R_i^j \): The entry in the routing table \( R \) at column \( i \) and row \( j \).
- \( L_i \): The \( i \)th closest node ID in the leaf set \( L \).
- \( D_i \): The value of the \( i \)th digit in the key \( \text{D} \).
- \( \text{shl}(A, B) \): The length of the prefix shared among \( A \) and \( B \), in digits.

### 3.2. Implementation details

We used Pastry API (Table 1) with some functions modified as described below. The following notation is used. By \( \{q\} \in \text{msg} \), it is meant that \( \text{msg} \) contains \( q \). The function names are in Italic. When a piece of information, such as \( \text{Inf} \), is added to a specific message, it is shown as: \( \text{newmsg} = \text{msg} \cup \{\text{Inf}\} \).

Content \( \text{X} \) in a node is converted to a 160-bit number using a hash function \( \text{HF} \). In FreePastry, each node is identified by a 20-byte number. Then the node sends a pointer containing the node’s ID to node \( T \) with the ID closest to \( \text{HF}(\text{X}) \).

\[
\text{routeMsg}(\text{HF}(\text{X}), \text{msg}, \text{cred}, \text{opt});
\text{\{HF}(\text{X}), \text{node ID}\} \in \text{msg}
\]

For more information about \( \text{cred} \) and \( \text{opt} \) arguments of \( \text{routeMsg} \) refer to [24]. Each node has a table with hash numbers close to its ID and the IDs of nodes that can provide the content. When a receiver is seeking senders with the desired content, it should run function Search which returns the IDs of the nodes that contain a specific media. \( \text{Search} \) can be implemented as follows.

\[
\text{nodeIDArr} = \text{Search(String name)}
\{
\text{routeMsg}(\text{HF}(\text{name}),\text{msg},\ldots);
\text{rmsg} = \text{SearchResultReceived(HP(\text{name}))};
\text{return getHosts(rmsg)};
\}
\]

\( \text{SearchResultReceived} \) is blocked until search result \( \text{rmsg} \), containing nodes’ ID having desired media, comes back from the node having ID close to the ID of desired node. It is worth noting that \( \text{rmsg} \) is sent by a node with an ID close to \( \text{HF}(\text{name}) \) and contains IDs of all nodes (extracted by \text{GetHosts}) that can provide \( \text{X} \). Node \( R \), interested in \( \text{X} \), sends a message to each of these nodes and asks them to send it their characteristic parameters such as bit-rate and availability:

\[
\text{for } j = 0 \text{ and } j < \text{length}(\text{nodeIDArr}),
\text{routeMsg(nodeIDArr}(j), \text{msg}, \text{cred}, \text{opt});
\{\text{receiverNodeID}\} \in \text{msg}
\]

Each sender, \( P \), sends it’s \( A_p \) and \( R_p \) to the receiver:

\[
\text{// In each sender:}
\text{routeMsg(receiverNodeID, msg, cred, opt);}
\{A_p, R_p\} \in \text{msg}
\]

### Table 1

<table>
<thead>
<tr>
<th>Returns</th>
<th>Function prototype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>EnrouteMessage (Message msg, Id key, NodeHandle nextHop, SendOptions opt)</td>
<td>Called by pastry when a message is enroute and is passing through this node</td>
</tr>
<tr>
<td>Void</td>
<td>messageForAppl(Message msg)</td>
<td>Called by pastray when a message arrives for this application</td>
</tr>
<tr>
<td>Void</td>
<td>routeMSG (Id key, Message msg, Credentials cred, SendOptions opt)</td>
<td>Route a message to the live node D with node ID numerically closest to key.</td>
</tr>
</tbody>
</table>
Each intermediate node responsible for routing msg from candidate senders to the receiver should add its characters to the msg. When a message is received by a router node, Pastry calls \texttt{enrouteMessage} function, which should be changed as follows:

```c
// In each router
enrouteMessage(Message msg, Id key, NodeHandle nextHop, SendOptions opt)
{
    ...
    newmsg = msg U \{routerID, availability, bitrate\};
    routeMSG(receiverNodeID,...);}
```

After receiving messages from the senders, the receiver constructs its multi-sender tree. Using a multi-sender algorithm, the receiver selects $k$ active senders (using a multi-sender algorithm) and informs them to start transmission. As mentioned before, each sender should be informed of \{$r_{\text{min}}, j\$} as well.

For node ID $\in \text{activeSenderSet}$ do

```c
routeMSG(node ID, msg, cred, opt);\{n, r_{\text{min}}, j\} \in \text{msg}
```

The receiver should also make a multicast tree table which determines intermediate routers between the receiver and each sender.

The next receiver should first run the \texttt{Search} function to find candidate senders. Then it has to send messages to each sender to request their characteristics. In addition to transmitting $A_p$ and $R_p$, the active senders give the IDs of the current receivers, which are queried by the new receiver for getting their multicast tree tables. Using all collected information, the new receiver generates its multi-sender multicast tree.

It is worth noting that we do not consider the network topology in the proposed algorithm. In fact, MSMC is completely run at the application level. Topology considerations are required and used by the multi-sender algorithm. For example, in FastIPROMISE [10], the multi-sender algorithm we employed, chooses a set of active senders due to the senders’ characteristics and the network specifications.

3.3. Distributed management

As mentioned before, we use some nodes as application layer routers to deliver data to the receivers. As nodes of a P2P network, these temporary senders may leave the network at any time. In this section, we discuss recovery of the multicast tree from such failures. Note that in the case of the failure of an original sender (as opposed to temporary ones = routers), the multi-sender algorithm should replace the failed sender by a stand-by candidate sender.

Consider the structure shown in Fig. 5. Suppose receiver $R$ is receiving data from $S$. Virtual router $Y$, on the path from $S$ to $R$, gets the stream from $Z$ and routes it to $R$. $Y$ is temporary sender for $T_1, T_2, \ldots, T_k$. We call $X$, the successor of $Y$, which is the father of $X$ as well as $T_1, T_2, \ldots, T_k$. $Y$’s father, $Z$, is the grandfather of $X$. $T_1, T_2, \ldots, T_k$ are the children of $Y$. Note that a successor receives the whole data which has been passed to its father. That is while a node may (and usually does) send only a portion of its received data to its children.

$X$ contains all information about $T_1, T_2, \ldots, T_k$ IDs as well as the portion of the stream each child takes from the father $Y$. $X$ also knows its grandfather’s ID. If $X$ receives no data from $Y$ within a certain period, it assumes that $Y$ is dead and sends a message to its grandfather. This message is routed by Pastry. As it is mentioned in Sections 3.1 and 3.2, a new node such as $Y'$ becomes father (i.e., $Z$ transmits data to $X$ through $Y'$). If more than one router is added to the path between $X$ and $Z$, the grandfather ID in $X$ should be corrected. $X$ informs $Y'$ about its newly adopted children and their portions. Thus, failure of $Y$ is dealt with in a distributed manner. If $Y'$ has less upload bandwidth than $Y$, its children should look for new resources.

In the case of simultaneous death of $X$ and $Y$, $Y'$, successor of $X$ (not shown in Fig. 5), who cannot reach its grandfather ($Y$), tells $R$ that it is an orphan. Upon receiving such message, $R$ makes a new path to $S$ and updates its routing table, which is also sent to the receivers that used the routers on the path from $S$ to $R$ as their temporary senders.
Although this reconstruction process may take long, low probability of the deaths of both a father and its successor justifies the method in practice.

4. MDC and buffering

This section explains a media splitting method which is an essential task for any multi-sender scheme.

An important concept in media streaming over network is buffering time, $t_0$, during which the media is received and buffered by the media player before starting playback. If $R_b$ denotes the stream bit-rate, $R_b t_0$ is the player’s buffer size. The larger the amount of $t_0$, the more robust is playing in events of network problems such as jitter, death of a sender, etc. The drawbacks of a large $t_0$ are the large buffer size and the longer delay in playback. Thus, finding the optimum $t_0$ for given network conditions is an important problem. Here, for simplicity, $t_0 = 1$, thus the bit-rate is equal to the buffer size.

Let us assume that the senders are using a standard video codec such as MPEG2. The output of such codec consists of a number of layers. The first layer (“the base layer”) is crucial for decoding while the others increase the quality of multimedia at the expense of a higher bit-rate. Suppose $M$ layers are sent to the receiver at a predetermined $R_b$ (Fig. 6). $R_i$ is the bit-rate required for playing layers 0 till $i$.

For every $k$ bits of data, one can generate $n - k$ parity bits such that for correct decoding only $k$ bits of the $n$ transmitted bits (and their positions) are required, using Reed–Solomon codes [25]. Because of their importance in the quality of decoded media, the lower layers of stream should be encoded more robustly. To this end, the $i$th layer is divided into $i + 1$ partitions and $R_S(n, n - i - 1)$ is computed for each partition [23]. Thus, each layer is converted to $n$ parts, $i + 1$ parts carry the data and the rest are recovery information (Fig. 8). One can easily verify that the data rate is

$$R_i = \sum_{j=0}^{N-1} \alpha_j R_j$$

where $\alpha_j = \frac{N}{(j-1)2^j}$, $\alpha_{N-1} = 1$

$R_i$ is the total bit-rate that the senders should provide, so that the receiver can play the media at $R_b$. By receiving $i$ descriptions, the playback quality is equivalent to layer $i$ (i.e., the more descriptions, the higher the quality of video). At a higher bit-rate, more descriptions can be delivered to the receiver, and the video quality is improved. Therefore, for simplicity, we plot the received bit-rate as an equivalent measure of quality.

Every sender has $n$ buffers, each with size of $R_i$ (assuming $t_0 = 1$). Each of the mentioned $n$ descriptors is fed to a specific buffer. For sender $i$, descriptions in buffers numbered $J_i$ are transmitted at rate of $r_i$.

For the second receiver, the contents of buffers $J_i$ of $i$th sender are put together and the above algorithm is run again to create $n$ descriptions that are fed to a new set of buffers. Then $P_j$ sends the contents of buffers numbered $K_j$ (Fig. 7).

Note that by increasing the number of receivers, $R_i$ for the last receiver increases as well. This is because of RS coding which sends parity for each description of the last receiver, even for its parity named as FEC (Forward Error Coding) in Fig. 8. Although such increase only linearly depends on the number of receivers, the network may be
overloaded in the case of a popular broadcast. To work around this problem, we may stop sending recovery information after joining of the first few receivers.

5. Simulation results

In our experiments, a FreePastry P2P network with 80 nodes is simulated on a single 3 GHz Intel PC using JVM (Java Virtual Machine) technology. Upload bandwidths (in kbps) are samples of \( U(300,600) \). Each node has an availability probability sampled from \( U(0,1) \) and its download bandwidth (in kbps) is a sample of \( U(300,800) \). \( U(X,Y) \) denotes the uniform distribution in \([X,Y]\). The experimental results reported by Bhagwan and others [26] showed that the availability of nodes in a short period of time (e.g., 10 hours) is uniformly distributed between [1]. Since video clips are usually much shorter than 10 hours, it is reasonable to assume that the probability of server availability is uniformly distributed between zero and one.

Each receiver uses FastIPROMISE [10] to choose the best set among all candidate senders.

In Fig. 9, the bit-rate provided to the second receiver over time using the proposed method (MSMC) is compared to that provided using the pure multi-sender algorithm of [8] run twice when 16 senders are contributing. It is observed that MSMC improves EBR of the second receiver by 30%, rendering the same service quality as that given to the first receiver (Section 4). Moreover, the bit-rate fluctuates less by using MSMC.

The effect of increasing the number of senders is shown in Fig. 10. Offered EBR to the second receiver in both algorithms is increased with the increased number of senders. MSMC outperforms MS again.

In Fig. 11a and b the histograms of the received bit-rate for the first three receivers are shown when MSMC algorithm and pure multi-sender algorithm of [8] are employed.

Fig. 9. The bit-rate offered to the second receiver using MSMC and multi-sender alone for 16 senders.

Fig. 10. The bit-rate provided to the second receiver when the number of senders is 32.

Fig. 11. Histogram of bit-rate provided to the first three receivers (a) using MSMC method (b) using pure PROMISE multi-sender algorithm.
It is observed that all three receivers get EBRs about 900 Kbit/s, when the proposed method is used. That is while, EBR is both reduced and spread (i.e., more likely to have EBRs considerably higher or lower than the average) for the second or the third receiver, using multi-sender only: the average EBR for the third receiver is less than half of the desired bit-rate.

Suppose \( n \) receivers exist. We define \( R_u \), the reuse parameter, as follows:

\[
R_u = \frac{\sum_{i \in \text{receivers}} BR_i}{\sum_{s \in \text{senders}} r_s}
\]

in which \( r_s \) is bit-rate provided by each active sender and \( BR_i \) is the bit-rate received by receiver \( i \). To interpret the concept of reuse parameter, note that \( \sum_{i \in \text{receivers}} r_s \) is the total bit-rate of all source nodes (a network resource) consumed by all receivers. \( \sum_{i \in \text{receivers}} BR_i \) is the amount of bit-rate provided to all receivers by all senders. Hence, \( R_u \) is an indication of network resource reuse.

For example suppose two receivers with desired rates of 1 Mbps. By using a pure MS (multi-sender only), for each receiver, we should supply some senders giving a total of 1 Mbps, giving \( R_u = 2 \times 1 = 1 \). By application of our MSMC method to the same problem, the second receiver is able to get all its data from virtual routers, making \( R_u = 2 \times 1 = 2 \). And a higher \( R_u \) means better usage of network resources.

Table 2 provides \( R_u \) measured values for different number of receivers in using MSMC method. Note that using a pure MS, \( R_u \) is always 1.

<table>
<thead>
<tr>
<th>Number of receivers</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_u )</td>
<td>1.95</td>
<td>2.85</td>
<td>3.8</td>
<td>4.77</td>
</tr>
</tbody>
</table>

Table 2

\( R_u \) values for different number of receivers using MSMC method

6. Conclusions

All existing multi-sender methods maximize the bit-rate provided to the first requesting node. When another node requests the media from the same senders, the algorithm must be run again. Since the limited senders’ bandwidths are already committed to the first receiver, the bit-rates offered to the next receivers are likely to be unacceptably low. A multi-sender algorithm tries to maximize the quality for a single receiver, and to that end, it uses up all good sources.

In this paper, we proposed a method that overcomes this problem by using temporary senders – the peers between the senders and the receivers that inevitably have parts of the streamed media. Our simulation results demonstrate that, using the proposed method, the bit-rate offered to the second peer is almost the same as that offered to the first receiver that is maximized by the underlying multi-sender algorithm. This is achieved at a minimal control overhead for the network.

Currently, the proposed method is transparent to the underlying multi-sender algorithm. Ideally, however, the proposed method can generate a shorter multicast tree (i.e., shorter delivery delay), should it collaborate with the multi-sender algorithm.

Also, the proposed method does not consider the availabilities and bit-rates of the virtual routers in sender selection. Developing a multi-sender algorithm which considers temporary senders characteristics is another avenue of future research.

Acknowledgement

The authors would like to thank Professor Mohsen Shiva for his kind support and accommodating us in the digital signal and image processing lab.

References


Keivan Ronasi received the BSc degree in Electrical Engineering from Isfahan University of Technology, Isfahan, Iran, in 2004, the MSc degree in Communication systems from the University of Tehran, Tehran, Iran, in 2007. His current research interest is Wireless Communications, Peer-to-Peer networks, and network management for ad-hoc and sensor networks. He is a student member of IEEE.

Mohammad Reza Pakravan graduated as a top ranking BSc student from University of Tehran in 1990. He obtained his MSc and PhD degree from University of Ottawa in 1994 and 2000 respectively. He worked in Nortel Networks for more than three years as a member of scientific staff. Dr. Pakravan joined Faculty of Electrical Engineering at Sharif University of Technology in 2001. His research interests are in the area of data networking protocols and algorithms, broadband wireless and wireline communications and applications of signal processing in telecommunication systems.

Alireza Nasiri Avanaki received his BS and MS in electronics and communication systems from Sharif University of Technology, Tehran, Iran, in 1995 and 1997. In 2004, he received his PhD in signal and image processing from University of British Columbia, Vancouver, Canada. Currently, he is an assistant professor at the ECE department of the University of Tehran, Tehran, Iran. His research interests include applied multimedia processing.

Mohammad Hamed Firooz received the BSc degree in Electrical Engineering from Isfahan University of Technology, Isfahan, Iran, in 2004, the MSc degree in Communication systems from the University of Tehran, Tehran, Iran, in 2007. His current research interest is Wireless Communications, Peer-to-Peer networks, and network management for ad-hoc and sensor networks. He is a student member of IEEE.

Mohammad Reza Pakravan