PPM - A Hybrid Push-Pull Mesh-Based Peer-to-Peer Live Video Streaming Protocol

Adel Ghanbari*, Hamid R. Rabiee*, Mohammad Khansari†, Mostafa Salehi*
*Digital Media Lab, Department of Computer Engineering,
Sharif University of Technology
Email: aghanbari@ce.sharif.edu, rabiee@sharif.edu and mostafa_salehi@ce.sharif.edu
†Faculty of New Sciences and Technologies, University of Tehran
Email: m.khansari@ut.ac.ir

***Abstract*—Using Peer-to-Peer (P2P) overlay networks have become a progressively popular approach for streaming live media over the Internet due to their deployment simplicity and scalability. In this paper, we propose a new hybrid push-pull live P2P video streaming protocol called PPM that combines the benefits of pull and push mechanisms for video delivery. Our main goal is to minimize the network end-to-end delay compared to the pure mesh networks. The PPM consists of two phases; Pull-based and Push-based. In the first phase, a new peer joins to the network based on a pull-based mechanism. In the second phase, a parent node based on the peers' overlay hop count in the mesh topology is selected. Then, a dynamic tree is constructed to push the high priority video frames to the children of the selected parent. Using OMNET++ as the simulation platform, we show that beside significant improvement on the end-to-end delay, PPM achieves lower visual distortion compared to the pure mesh networks. Moreover, the simulation results confirm superiority of the PPM in comparison with the popular mesh-based P2P streaming systems.**

***Index Terms* – Peer-to-Peer live streaming, push-pull protocol, mesh-based networks.**

I. Introduction

By the advent of accessible Internet connection, the demands for online video are continuously increasing. TV broadcasting suppliers deploy new ways to provide the customers with streaming video over the Internet. Up to now, several video distribution models such as IP multicasting and Application Layer Multicasting (ALM) have been proposed [1]. Among the ALM models, Peer-to-Peer (P2P) streaming systems have attracted more researches in the field of design and implementation because of their scalability and deployment simplicity. P2P streaming systems are applied for both video On-demand that is considered as stored video, e.g., Bittorent [2] and Live video streaming, e.g., PPLive [3]. Regardless of their applications, generally, P2P streaming systems are classified into three categories; 1) Tree-based [1] [4] [5] that are known as structured overlay architecture in which the video is pushed from the origin server (root node) to its children. These systems are a well-organized structures and have simple implementation mechanism. 2) Mesh-based [6] [7] where peers are not limited to a static topology and they pull the content from their neighbors by sending request messages to them. The main advantage of the mesh-based systems is that they are very robust to peer churn. And 3) the Hybrid architectures [8] [9] that deploy both tree and mesh topologies in their delivery system. It means that, a push-pull approach is deployed on the hybrid system for delivering the video content. The main advantage of designing hybrid structure is to exploit the benefits of both tree and mesh structures and achieve the quality of service standards.

In the hybrid structures, a separate control topology is needed for both tree and mesh topologies that leads to high control overhead [9] [10]. In addition, using tree and mesh topologies in the hybrid structures have some disadvantages. Using tree, is vulnerable to nodes dynamic and churn; whenever a new node either wants to join or leave the streaming session, the whole tree should be reconstructed. Additionally, tree-based systems suffer from single point of failure, bandwidth utilization problems. Moreover, they need a controlling mechanism or algorithm to control the tree topology. Even though, the mesh-based systems mitigate the bandwidth utilization problem of tree systems, but they suffer from high controlling overhead that is due to exchange of controlling messages such as request/response and buffermap messages. Furthermore, different data packets may pass over different routes, as a result, users may suffer from video playback quality degradation [7] ranging from low video bit rate and long startup delays. Hybrid systems although, improve the resiliency of the network, their construction and maintenance are complex [8].

In this paper, we propose a new protocol with a hybrid push-pull approach, called Push-Pull Mesh (PPM), which is deployed on a pure mesh-based network. We have used single layer video coding and introduced a new priority-based delivery scheduling mechanism. This new scheduling mechanism is exploited to emulate the tree behavior. In this mechanism an overlay hop count-based parent selection method is developed to select a parent among nodes' neighbors; then a dynamic tree within the pure mesh topology will be constructed. Afterward, the video packets with the higher priority are pushed through that tree.

Consequently, we organize our push-pull protocol in a pure mesh-based network that does not need any extra controlling mechanism and its construction is not complex. Likewise, because the single layer video is used in our video delivery system, there is no need of any knowledge of video in the scheduling section. We show that by omitting some of exchanged messages between peers, the lower end-to-end delay and also lower overhead can be achieved. In addition, the push mechanism introduced in the PPM, causes significant decrease in video distortion, while it delivers the video packets to the destination, before their playback time is expired. The simulation results show that we could achieve 15% and 77% decreases in the end-to-end delay and video distortion, respectively. The results also indicate that we could achieve our design goals which are handling single point of failure that occurs in tree-based systems and also lower controlling overhead.

The rest of the paper is organized as follows. In section II, we discuss some newly introduced hybrid protocols as related work to our proposed protocol. The details of the proposed protocol are explained in section III. In section IV, the simulation results are discussed. Finally, the conclusions and future plans are presented in section V.

II. Related Work

Generally, push-pull video streaming protocols can be achieved through two approaches. The first approach, called mesh-tree, is to construct a hybrid topology by combining tree and mesh structures [11] [8] [10]. The second approach, called mesh-based, is to alter the mesh topology and deploy a push-pull scheme [12] [13]. In the following, we survey the state of the art push-pull P2P video streaming systems in these two categories.

GridMedia [11] is an unstructured Mesh-Tree P2P protocol for live media streaming employing a push-pull mechanism. In this protocol, each node works in pure pull mode in the first time when connecting to the network. After that, based on the traffic from each neighbor, the node will subscribe the pushing packets (which include packets that should be pushed) from its neighbors at the end of each time interval.The proposed push-pull mechanism in GridMedia reduces the latency and inherits good features such as simplicity and robustness of pure pull method. As the GridMedia uses multiple description coding, it causes complex construction of mesh-tree architecture.

In the mTreebone [8], the main idea is to identify some stable nodes in mesh-based. Then video is pushed through the tree that is constructed by the root node and these stable nodes. The mesh topology then is used to help nodes to obtain the missing video blocks in the network. mTreebone uses Multiple Description Coding (MDC) for its video delivery system. The main drawback of the mTreebone is that it is not resilience enough in presence of flash crowded situation.

In [12] an Interleave protocol has been proposed that deploy the advantages of the push-pull mechanism on an unstructured Mesh-based network. This protocol interleaves push and pull mechanisms in different time intervals, and a peer is only in one of push or pull state. When the peer is in the push state it will select a neighbor and pushes the video chunks with the highest sequence number. When the state of the peer is pull, if it has available bandwidth and its attempts are less than the pull attempts threshold, then it will send a pull message to one of its neighbors.

As mentioned above, the Interleave protocol has cyclic behaviors that causes non-concurrent push and pull mechanisms. In addition, the protocol works in the situation that there is no need to have any information about available chunks of neighbors. Therefore, most of the network bandwidth will be overloaded due to exchanging useless push and pull messages. Moreover, the Interleave protocol has an impractical assumption which is all nodes have equal and unlimited upload/download bandwidth.

In the New CoolStreaming [13], as a Mesh-based protocol, the video is divided into N sub-streams without any coding techniques. These sub-streams contain the video block and all nodes can request each sub-stream separately. The push mechanism in the New CoolStreaming works in the way that the information about sub-streams and their contents (video blocks) is carried on the nodes buffermap that is exchanged between them periodically. Each node based on the missing video block, requests that block within a single pull request. Then, the provider will push the sub-stream that contains that video block to the node. Overhead reduction and time saving during block transmission from the sender to the receiver are the most significant advantages of the New CoolStreaming protocol. But the sub-streaming that used in this protocol increases its construction complexity.

A hierarchical push-pull and Multi-Tree based scheme is proposed in [10] that contains two topologies: control topology and multi-source multicast tree. The multi-source multicast tree is built above the control topology. The control topology is used to control and manage membership, to group closer peers together, and to simplify the selection of multi-source data paths. The multi-source multicast tree is considered to decrease the impression of node churn and to resolve the problems, like potential bandwidth bottlenecks.In this protocol, peers are arranged into different clusters in a hierarchical structure. The number of nodes in each cluster is limited by [k, 3k-1], where k is a universal constant predefined in this protocol. After join procedure, the newly joined peer can easily place [k, 3k-1] peers as its strong links and another [k, 3k-1] peers as its weak links. The newly joined peer then selects N parents from both of its strong and weak links for push-type delivery. If the packet is lost in the push-type path, the packet will be pulled from other links. The multi-source multicast tree makes the system adaptive to node churn and packet loss. A separate control topology leads to high overhead in this protocol. Consequently, the efficiency of this system will decrease. In addition deploying multi-tree in delivery topology makes the topology construction complex.

Regarding the discussion above, although some of protocols are mesh-based protocols, they deploy MDC or sub-streaming coding that require a complex usage and implementation compared to single layer coding. On the other hand, those who used single layer video coding, they are constructed based on multi-tree topology that has reconstruction complexity problem [8]. Except the Interleave protocol that has the problem of asynchronous push and pull mechanism. But our protocol, the PPM, is deployed on the pure mesh-based network. It uses the single layer video coding, so, it has simple usage and deployment and also it deploys push and pull mechanisms concurrently.

III. Proposed Method: the PPM

Our proposed protocol focuses on a pure mesh topology and uses a single layer video coding in its video delivery system. We introduce a new priority-based video delivery scheduling mechanism for the scheduling method in which all peers request the video packets based on the frame types. Then, an overlay hop count-based parent selection method is used to construct a dynamic tree within the pure mesh topology. After that, the most important video frames that have higher priority rather than the other frames in a Group of Picture (GoP), will be pushed through the constructed dynamic tree.

Algorithm 1: PPM's Scheduling Algorithm

```
Input:
$isChild$: The state when peer is child
$isParent$: The state when peer is parent
$playingDeadline$: Playing time deadline for a frame
$missingFrame$: The missing frame in the node's buffer
$F$: A frame type variable
$pushingFrame$: I or P frames that must be pushed
$childList[i]$: An array consists of a parent's children

Scheduling:
if $isChild$ then                                    //Pull-based phase
    if $missingFrame == I \ or \ P$ then
        if $playingDeadline \leq 2 \times RTT$ then
            $F \leftarrow missingFrame$;
            Pull($F$);
        end
    else
        Wait-for-Push();
    end
end
if $isParent$ then                                   //Push-based phase
    for every $childList[i]$ do
        Push ($pushingFrames$);
    end
end
```

Generally, the PPM scheduling is developed into two phases (Alg.1): the pull-based and the push-based phases. Each node, in the startup, works under the pull-based phase. The parent selection and push method are invoked in the push-based phase. The following sub-sections explain the protocol mechanism in detail.

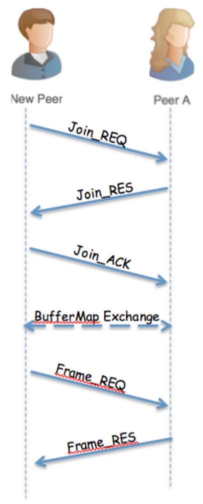


Fig. 1. Application level join process in the overlay mesh network

A. Pull-based phase

When a peer joins the network, firstly, it contacts a node so called Tracker that has global information about the existing peers in the network. It requests a random number between 3 to 5 [14](by sending a Neighbor_Request message) that indicates it requires some neighbors. The tracker sends back a Neighbor_Response message in which the addresses of some peers corresponding to the requested number are resided. The tracker returns the list of neighbors that are close to each other from the aspect of their entered time to the network.

After retrieving the neighbors' address the newly joined peer contacts each of them and sends a Join_REQ message to inform them that it wants to make a connection. On the opposite side, if the peer has a free position for the nodes request it sends a Join_RES message to the peer; otherwise, a Join_DENY message will be sent back. When the newly joined peer receives the accept message, it replies with a Join_ACK message to confirm the neighboring connection between itself and the opposite peer. (Fig.1)

If a peer received a Join_DENY message from a peer, it will contact with the Tracker and request another neighbor and obtains a new neighbor's address and the join process will be recalled again. After establishing the connection between two nodes, they start to exchange their Buffermaps.

As Fig.2 shows, the buffermap is a map that shows the available video frames in the node's buffer. When a video frame is available in a cell of the nodes buffer, its corresponding bit in the buffermap's cell is 1, otherwise, the value of the buffermap's cell is 0.

When the newly joined peer receives the first buffermap message from its neighbors, since we deploy the live streaming, it should shift its playing window and synch with its neighbors. After that, it randomly selects one of its neighbors and requests the video packets. If the requested frame does not arrive within a time less than the Round Trip Time (RTT) seconds, the peer will repeat its request by randomly choosing

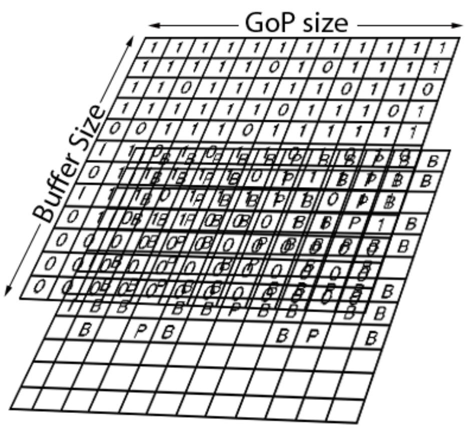


Fig. 2. Buffermap concept

another neighbor.

B. Push-based phase

To achieve our goal that is deploying the push mechanism on a pure pull-based mesh topology, we have introduced a new priority-based scheduling mechanism in our protocol.

We have used the typical MPEG compression standard and introduced the GoP structure. In a GoP, there exists 1 I frame, 3 P frames, and 8 B frames. P frames are called inter-coded, and they can be predicted from their previous P or I frames. B frames are bidirectional and predicted from previous and future P or I frames [15] (Fig.3). These frame types information only used for deploying the push mechanism.

By considering these principles, if the base frame of a GoP (I frame) is lost, then all the other received frames of corresponding GoP cannot be decoded. This condition is also true for the first P frame of each GoP.

Regarding the characteristics of MPEG standard, we assign a priority to each frame in the GoP. I frame has the highest priority in a GoP. After I frame, the first P frame has the priority in the second level. We call this frame, P1 frame. Based on the principle of each GoP, if a base P1 frame is lost, the following correctly received frames will be lost. We exploit this priority to make our push mechanism possible to work correctly in the network.

To push the most important frames on the mesh network, we have to choose a proper node to act as a parent and push those frames to its children. On the other hand, when a push procedure is going to appear, it is very important to prevent from data redundancy. To address this problem, we introduce a parent selection function in which the best potential node from the aspect of shortest distance from the server will be selected. We select based on minimum overlay hop count far from the video server. The minimum overlay hop count helps us to make sure that a node that can provide us with lower end-to-end delay is selected and it can deliver the network packets faster. The overlay hop count equals to the number of peers that a network packet passes as a path until it reaches the destination peer. After selecting the appropriate node, the child sends the Push_Req message to the target parent.

On the other hand, when a node receives a Push_Req message, if it has free position to accept a child, it replies with a Push_Accept message; otherwise, it replies with a Push_Deny message to inform the child that it has no available position for the sender of the request.

We have defined a State parameter for each node in the PPM. Typically, when a node joins at the Initialize phase, its state will be set to REGULAR. When it is accepted to be child, its state will be changed to CHILD. If a node receives a Push_Accept message, it will finalize the parent selection by changing the state of the sender of this message from REGULAR to PARENT to show that its parent is the peer that has answered with Push_Accept message. At last, if a node encountered with a Push_Deny message, it changes the state of the opposite peer from REGULAR to PUSH_DENIED; therefore, it should recall the parent selection method again to select the parent from its remaining REGULAR neighbors.

Although, we do not focus on resource allocation in our design, we used a simple upload bandwidth allocation technique for each parent to select child nodes. In this technique, each parent node, dedicates a minimum amount of upload bandwidth to each of its children. Therefore, based on parent's uplink, we divide it into multiples of 256 kbps (due the video bit rate of 256 kbps). For example, if the node's uplink is 1 Mbps, there exists 4×256 kbps slices. Then we reserve two slice (about 50% of upload bandwidth) for neighbors in pull phase that might request the missing chunk (such as B, P or I frames). And the remaining two slices (remaining 50% of upload bandwidth) will be dedicated to child nodes for push phase. Hence the node can select at most two nodes as children. It is obvious that this technique is not an optimal resource allocation solution, but we insured that the parent could serve the child nodes.

1) Parent selection procedure: We have considered approximation of nodes overlay hop count in the network to select an appropriate parent. Each node calculates and puts its overlay hop count in its buffermap and exchanges that between its neighbors periodically. We exchange the overlay hop count periodically because we want to have the chance of parent selection at any time of streaming session. For example, in presence of churn or parent failure, a child node can easily make its best decision and choose another appropriate parent. Therefore, when a peer wants to select the parent, it compares

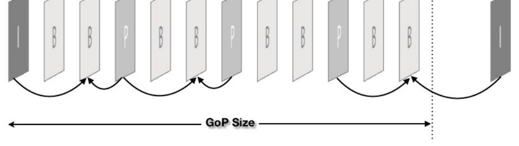


Fig. 3. Typical GoP structure in MPEG standard

its neighbors hop counts and chooses the neighbor with the minimum hop count among the neighbors. Then it sends its Push_Req message and the procedure mentioned in subsection *B* (the push-based phase) will be done.

The parent selection method has some key points in the PPM. Firstly, We emphasize that, by parent selection based on minimum overlay hop count, the nearest node to the origin server will be selected. The nearest neighbor node to the server has this advantage that the node can deliver the video frames faster due to its distance from the server. Secondly, it should be noticed that, when a peer starts playing the video, the parent selection method would be called. Considering this point for evoking the parent selection method guarantees this fact that, the node has the sufficient number of neighbors to select a parent from them. Nodes in the PPM start their playback if they get all their neighbors. In addition, playing moment is an appropriate time to push the available packets to the children nodes. It means that there are sufficient amounts of video packets in the nodes buffer for pushing them to the children. Thirdly, using parent selection method leads to construction of a dynamic tree above the mesh topology without maintaining additional topology and data beside the base mesh network. And finally, as only one parent is selected for each child, so it does not receive similar pushing packets from different neighbors.

2) Overlay hop count calculation: We have added an additional field to each nodes video frame as Hop. The value of the hop in the server is set to 0. Since, a frame travels through the network and is obtained by every node, this value increases by 1. Therefore, each node receives different frames from various paths. To calculate the overlay hop count in each node, it maintains the latest 20 recently received frames hop. We estimate the overlay hop count equal to the average of these twenty hops. The value of the overlay hop count is continuously updated by receiving new frames from neighbors. Hence, whenever a peer wants to select its parent, it can easily choose the updated information.

3) Push mechanism: After parent selection process, whenever a parent node receives a frame, if it is P1 or I frame, it will push that frame to its children.

On the other hand, in order to prevent data redundancy, if a peer has a parent, it does not ask the frames whether their type is P1 or I. In the situation that the playing deadline for P1 frame or I is going to pass, the child requests that frame to prevent frame loss. Further explain, we have considered that if the frame didn't receive within $2 \times RTT$ seconds before its playing time, the peer will request that from its neighbors. In the case of parent failure or churn, its child nodes will recall the parent selection method again and choose another appropriate parent.

IV. Simulation and Results

We use OMNET++ [16] as the base platform, which is a discrete event and modular simulation tools for simulating various communication networks. Beside the OMNET++, for simulating the network level, physical and transport protocols the INET Framework [13] is used. On the top of the network, the overlay network is constructed by OverSim [17] which works based on OMNET++ and the INET framework. The OverSim is a framework that is used for simulating the overlay networks.

PPM simulation framework has three layers in which different parts of network are implemented: Underlay tier, Overlay tier and Application tier.

Each layer is considered as a module and should be implemented separately. As shown in Fig.4, the application and overlay tiers are implemented in Oversim and the underlay tier is performed via INET Framework.

To setup the simulation environment, we have used Star War IV trace file that can be downloaded from [18], which is MPEG-4 video coded and containing the Variable Bit Rate (VBR) video sequence with the average rate of $256kbps$. We have also randomly assigned the bandwidth between $512Kbps$ to $1.5Mbps$, based on the assumption in [19],to the peers to demonstrate that our network supports peers with heterogeneous bandwidth. Each peer in the network gets a random number of neighbors, which is between 3 to 5 [14]. We evaluate the PPM performance for 50 to 400 peers and ran each scenario for 200 seconds simulation time. Other simulation parameters are listed in Table I.

TABLE I
Simulation Parameters

Simulation Parameter	**Value**
Video codec	MPEG-4
Video FPS	25 fps
Number of frames in a GoP	12
Selected trace file	Star wars IV
Average video bit rate	256 kbps
Number of neighbors	Random (3∼5)
Nodes bandwidth	Random (512 kbps∼1.5 Mbps)

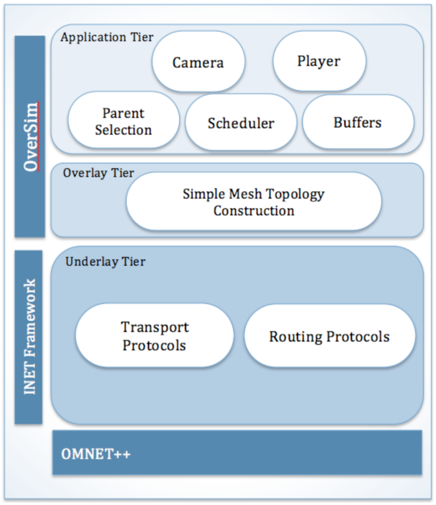
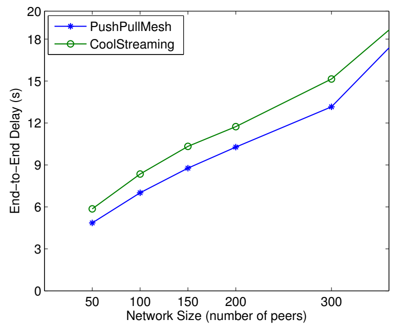
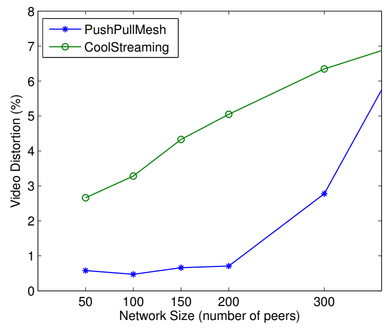


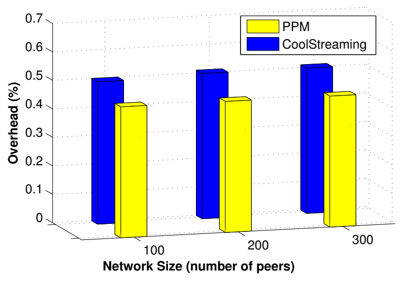
Fig. 4. PPM's simulation framework



(a) PPM's vs. CoolStreaming end-to-end delay



(b) PPM's vs CoolStreaming's video distortion



(c) PPM's vs. CoolStreaming's overhead

Fig. 5. Pperformance metrics

A. Performance metrics

To evaluate the performance of the PPM we compared it with an implemented CoolStreaming-Like protocol [6] that is a mesh-based network. It is mentionable that since the push and pull phases in PPM are done at same time and concurrently, we do not compare it with the Interleave protocol discussed in the related work. Our performance metrics consist of *end-to-end delay*, *video distortion* , *startup delay* and *control overhead*. Actually, these metrics are closely related to the quality of service. In other words, better results obtained regarding to these metrics, more quality of service will be reached. Hence, the users will receive the video with the acceptable quality.

1) End-to-end delay: The end-to-end delay is defined as the time that a frame takes to travel from the server to a peer. As depicted in Fig.5(a), the end-to-end delay increases as the network size grows; this is because, by increasing the number of peers (the network size), the distance between the source and the destination increases.

The end-to-end delay comparison between our protocol and the CoolStreaming shows that the delay produced in the PPM is about 15% in average lower than the CoolStreaming because the push method reduces the number of request/response messaging procedures; hence, the end-to-end delay decreases.

2) Video distortion: The video distortion is defined as the ratio of lost frames over the total number of frames streamed in the network. This parameter can be obtained from the following Equ.1:

$$Distortion = (1 - \frac{Total\_Number\_of\_Received\_Frames}{Total\_Number\_o\_Frames}) \tag{1}$$

After the simulation, the results show that the superiority of the protocol in comparison to the CoolStreaming protocol. As it can be inferred from Fig.5(b), for all numbers of peers that we test for our protocol, the video distortion is approximately 77% in average lower than the CoolStreaming.

The reason behind the decrement in the video distortion in comparison to the CoolStreaming is that the push method prevents frames from late arrival loss that causes the frame to be useless for playing. Similar to the end-to-end delay the distortion increases when the network scale increases.

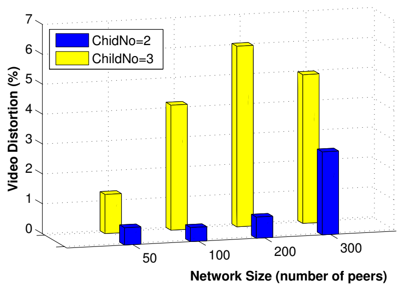
3) Startup delay: The time between connecting to the mesh network and starting the video playback is called startup delay. Another result observed from the simulation is that all nodes in both the PPM and the CoolStreaming start with the same startup delay. When the number of peers increases, it takes less time to start playing the video. Each node starts its playback whenever it is connected to all of its neighbors so when a peer is connected to the network with a larger number of peers, it can connect to its neighbors faster and obtain the video frames; hence, the startup delay decreases.

The startup delay increases suddenly from the network size of 50 to 100. This is because in the network size of 50, the nodes' number is few enough so that all peers are closed to the video server from the aspect of distance and obtain the video frames very fast. But the startup delay for the network size of 100 to 150 still increases rapidly because nodes distance increases from the origin server so peers start their playback with larger delay. By increasing the network size up to 300 or 400 peers, although the peers distance increases, the content availability also grows and more data will be injected to the network. Consequently, peers can obtain the video content faster and the startup delay decreases. We obtained from the simulation results that the PPM is getting a faster startup and lower delay the prioritized frame delivery.

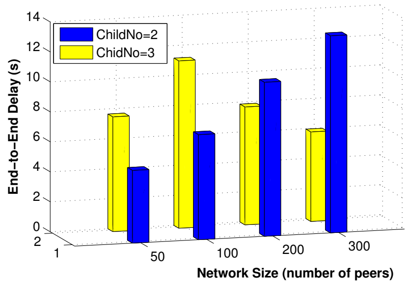
4) Control overhead: In each communication system, especially in P2P video streaming systems, in addition to video data exchange, a considerable controlling messages, such as buffermap messages, request and response messages, are exchanged between every two nodes. Hence, it imposes controlling overhead on the network. One of the fundamental goals of designing the PPM is possibly, reducing the amount of the controlling overhead. So based on the Equ.2 the PPM s overhead equals to the ratio of the amount of controlling messages (D_c) in packets, to the total amount data that peer received which is the sum of controlling data and the actual video blocks ($D_{total} = D_c + D_{actual}$).

$$Overhead = \frac{D_c}{D_{total}} \tag{2}$$

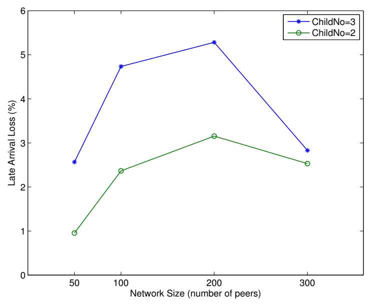
The Fig.5(c) indicates that in our fundamental design configuration we could reduce the amount of PPM's about 6% in



(a) PPM's video distortion for different values of child number

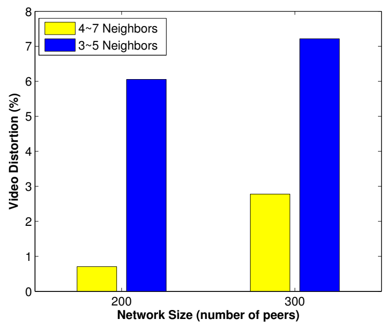


(b) PPM end-to-end delay for different values of child number

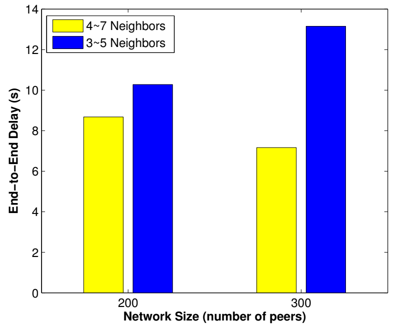


(c) PPM's late arrival loss for different values of child number

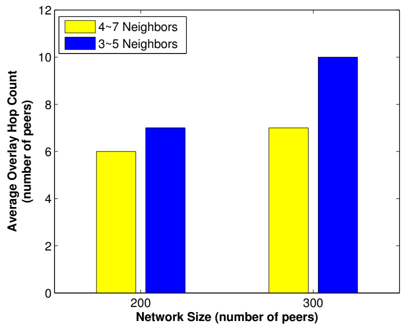
Fig. 6. Child number effect on PPM's performance



(a) PPM's distortion for different neighbors number



(b) PPM's end-to-end delay for different neighbors number



(c) Average overlay hop count for different number of neighbors

Fig. 7. Child number effect on PPM's performance

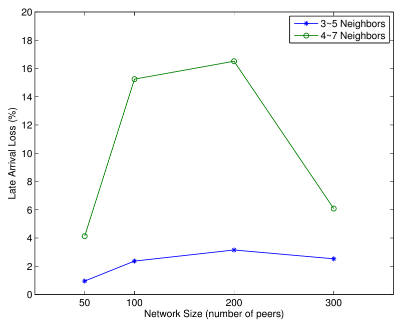


Fig. 8. PPM's late arrival loss for different number of neighbors

comparison with the CoolStreaming protocol. This is because by pushing the video frames by parent nodes, we have caused to prevent sending *Frame_Req* message from the child nodes to the parent nodes. Therefore it reduces the amount of exchanged controlling messages.

B. PPM's evaluation

To examine the capabilities of the PPM we have defined two scenarios in which the effect of child numbers and also the number of neighbors on the performance metrics are investigated. It should be mentioned that the simulation parameters are the same as those mentioned in Table I.

1) The effect of child number on the performance metrics: In this scenario we investigate the effect of children numbers regardless of used resource allocation technique. As Fig.6(a) shows, by increasing the child number (the number of a peers children) value from 2 to 3, the video distortion increases. This is because we have designed the PPM by adding minimum modification to the mesh network without considering any optimal resource allocation techniques. Therefore, when the number of children increases, the amount of required bandwidth for serving both child nodes and regular neighbors will increase. Hence, the lake of bandwidth causes an increase in the video distortion percentage.

Further more, increasing the child numbers value can help us to decrease the amount of end-to-end delay. Fig.6(b) proves this fact. As seen in this figure, the end-to-end delay decreases for the network size of 200 and 300 but does not for the size of 50 and 100. We justify this issue by saying that for the peers number of 50 or 100, as the network size is not big enough for the PPM to select a best neighbor as the parent node and the 3 child nodes may not be selected completely; so the child number of 3 cannot help us to decrease the end-to-end delay value; but for the peers number of 200 and 300, the

PPM parent selection and consequently child selection works successfully.

In addition it may raise another question that why does the end-to-end delay decrease while the video distortion increases? To answer this question, the main reason of distortion should be revealed in detail. We referred to the *Late Arrival Loss Distortion* measure and saw that the amount of this measure for the ChildNo = 3 is more than when have ChildNo = 2 where the ChildNo corresponds to the child number (Fig.6(c)). It means that the frames arrive after their playback deadline so they become lost.

On the other hand, the lost frames do not contribute in the end-to-end delay estimation. In other words, as the definition of the end-to-end delay implies, it corresponds to the timestamp when the frame is produced in the source server minus the timestamp when it is received in the destination peer. Thus, the frames lost in the network do not have any roles in the calculation of the end-to-end delay. Consequently, the amount of the end-to-end delay may increase or decrease in the network. Furthermore, as the video distortion increases, it can be inferred from Fig.6(a) and Fig.6(c) that the main reason for the distortion to happen is the late arrival loss that is the consequence of network congestion. For example, suppose that a peer with a low bandwidth resides at the up levels in the network. Hence, it cannot serve its neighbors correctly, therefore the network becomes congested. Then, the late arrival loss occurs and consequently, the video distortion increases.

2) The effect of neighbors number on the PPMs performance metrics: In this scenario we test the PPMs performance from the aspect of peers neighbor number. Typically, to compare with the mesh network, we considered the neighbor number to be a random number between 3 to 5 peers. We examined the neighbor number to be a random number between 4 to 7 peers and observed the results.

The results emphasize the fact that by increasing the neighbors number, the video distortion decreases (Fig.7(a)). This is because the overlay distance between peers decreases so the end-to-end delay also decreases. Fig.7(c) confirms our claim by showing the average overlay hop count for different number of neighbors. The figure shows that when the number of neighbors increases, peers are become closer, so the amount of overlay hop count decreases. Then we expect to obtain lower end-to-end delay that can be seen in the Fig.7(b).

Although, the end-to-end delay has a better situation in this scenario, the video distortion increases. This is due to the lake of bandwidth and the network congestion that prevents the peers from serving their neighbors. By the same justification, because some frames are lost due to late arrival loss and also only the correctly received frames are participate in the calculation of the end-to-end delay, the main reason of distortion in the network is the network congestion. Therefore, the video distortion increases while the end-to-end delay is decreased.

V. Conclusion and Future Work

In this paper we proposed a hybrid push-pull mesh-based P2P live video streaming protocol in which, by exploiting a dynamic tree, the most important video frames are pushed to the child nodes. The key role of the dynamic tree is that it omits some request/response messages in the scheduling part of the PPM. Consequently, some frames are retrieved with lower latency. In other words, the end-to-end delay and the controlling overhead decreased considerably. The simulation results showed that beside the improvement on the end-to-end delay and controlling overhead, PPM achieved lower visual distortion compared to pure mesh network.

Our work suggests several promising directions for further research. One direction is to study scalability and churn handling mechanisms to see how much the PPM is resilient when the number of peers becomes large. Another direction is to study optimal resource allocation techniques in parent nodes. And final direction can be the study of the effects of various mechanisms for parent selection method and finding optimal solution.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, “Scalable application layer multicast,” in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM ’02. ACM, 2002, pp. 205–217.

[2] (2012, Jan.) Bittprent. [Online]. Available: http://www.bittorrent.com

[3] (2012, Jan.) Pplive. [Online]. Available: www.synacast.com/en/

[4] J. Liu, S. Rao, B. Li, and H. Zhang, “Opportunities and challenges of peer-to-peer internet video broadcast,” vol. 96, no. 1, Jan. 2008, pp. 11–24.

[5] Y. hua Chu, S. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1456– 1471, Oct. 2002.

[6] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, “Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming,” in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, 2005, pp. 2102–2111.

[7] Y. Liu, Y. Guo, and C. Liang, “A survey on peer-to-peer video streaming systems,” *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, Jan. 2008. [Online]. Available: http://www.springerlink.com/index/10.1007/s12083-007-0006-y

[8] F. Wang, Y. Xiong, and J. Liu, “mtreebone: A collaborative tree-mesh overlay network for multicast video streaming,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 379–392, 2010.

[9] M. Moshref, R. Motamedi, H. Rabiee, and M. Khansari, “Layeredcast - a hybrid peer-to-peer live layered video streaming protocol,” in *Telecommunications (IST), 2010 5th International Symposium on*, Dec. 2010, pp. 663 –668.

[10] V. Gau, Y.-H. Wang, and J.-N. Hwang, “A hierarchical push-pull scheme for peer-to-peer live streaming,” in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 2066 –2069.

[11] L. Zhao, J.-G. Luo, M. Zhang, W.-J. Fu, J. Luo, Y.-F. Zhang, and S.-Q. Yang, “Gridmedia: A practical peer-to-peer based live video streaming system,” in *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, Nov. 2005, pp. 1–4.

[12] R. Lo Cigno, A. Russo, and D. Carra, “On some fundamental properties of p2p push/pull protocols,” in *Communications and Electronics, 2008. ICCE 2008. Second International Conference on*, 2008, pp. 67–73.

[13] B. Li, S. Xie, Y. Qu, G. Keung, C. Lin, J. Liu, and X. Zhang, “Inside the new coolstreaming: Principles, measurements and performance implications,” in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 1031 –1039.

[14] Z. Ren, J. Liu, F. Qin, and Y. Wang, “A survey on peer-to-peer streaming system’s neighbor number,” in *Computer Science and Engineering, 2009. WCSE ’09. Second International Workshop on*, vol. 1, 2009, pp. 421 –424.

[15] B. Akbari, H. Rabiee, and M. Ghanbari, “An optimal discrete rate allocation for overlay video multicasting,” *Computer Communications*, vol. 31, no. 3, pp. 551– 562, 2008, ¡ce:title¿Special Issue: Disruptive networking with peer-to-peer systems¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366407003209

[16] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, ser. Simutools ’08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1416222.1416290

[17] I. Baumgart, B. Heep, and S. Krause, “Oversim: A flexible overlay network simulation framework,” in *IEEE Global Internet Symposium, 2007*, May 2007, pp. 79 –84.

[18] (2011, Dec.) Video trace library. [Online]. Available: http://trace.eas.asu.edu/

[19] Z. Liu, Y. Shen, K. Ross, S. Panwar, and Y. Wang, “Layerp2p: Using layered video chunks in p2p live streaming,” *Multimedia, IEEE Transactions on*, vol. 11, no. 7, pp. 1340 –1352, 2009.