Multimed Tools Appl (2016) 75:17097–17119
DOI 10.1007/s11042-015-2979-5

Ppssm:push/pull smooth video streaming multicast protocol design and implementation for an overlay network

T. Ruso[1] · **C. Chellappan**[2] · **P. Sivasankar**[3]

Received: 31 July 2014 / Revised: 27 September 2015 / Accepted: 2 October 2015 /
Published online: 29 October 2015

Abstract IP multicast is one of the best techniques for video streaming on the Internet. It faces issues with respect to address allocation, routing, authorization, group management, security, and scalability. By default, local Internet Service Providers did not enable IP multicast services, because of the cost incurred in using multicast-enabled routers. To solve these issues some of the IP layer functionalities have been shifted to the Application Layer, thus leading to Application Layer Multicast (ALM) protocols. However, ALM protocols face issues related to synchronous data delivery, scalability, link stress, link stretch and node failures. Some of the existing protocols are CoolStreaming, and mTreebone. A novel ALM protocol based Push/Pull Smooth video Streaming Multicast (PPSSM) protocol is proposed in this paper, to increase the throughput and reduce the packet loss rate. The PPSSM protocol involves three stages, such as tree-mesh construction, dynamic buffer management and network coding techniques. In the tree-mesh construction, a tree consists of stable nodes and a mesh consists of unstable nodes. The proposed PPSSM optimizes the stable nodes in the tree, which minimizes or eliminates the pull operations from the unstable mesh overlay nodes, by exploring the potential of the stable nodes. Dynamic buffer management is achieved by setting the optimal buffer threshold value, using the optimization of the sensitivity parameters, such as packet loss and packet workload/delay by

✉ T. Ruso
racingruso@gmail.com

C. Chellappan
drcc@annauniv.edu

P. Sivasankar
siva_sankar123p@yahoo.com

[1] Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha University, Chennai 77, India

[2] GKM College of Engineering and Technology, Chennai, India

[3] Electronics Engineering Department, NITTTR, Chennai 113, India

the Infinitesimal Perturbation Analysis and Stochastic Approximation algorithms. In addition to the tree-mesh construction and buffer management, the introduction of the network coding technique will enhance the throughput and minimize the packet loss and delay. Finally, the performance of the proposed PPSSM protocol is compared with those of CoolStreaming, and mTreebone, and it shows improvement in respect of throughput, packet loss, and average decoding time.

Keywords Application layer multicast · Buffer management · Network coding · Load balancing

1 Introduction

The IP (Internet Protocol) multicast protocol [6] is an efficient protocol for one-to-many and many-to-many applications, viz., Internet Protocol Television (IPTV), Video Conference, e-Learning, Interactive Online video gaming, Virtual laboratory, Tele medicine, etc. The popular IP multicast routing protocol is Protocol Independent Multicasting (PIM). It has two variants: PIM-sparse mode [9] and PIM-dense mode [1]. These two protocols work with the Internet Group Management Protocol (IGMP) [7].Due to the lack of widespread IP Multicast support, researchers shifted the IP multicast functionalities to the application layer. In the application layer, the overlay network is formed through the software programs. An overlay network is a computer network, which is built on top of the physical network. The nodes in the overlay are connected through the logical or virtual links, each path of which is connected through many underlying physical links. The overlay network faces challenges with respect to delay, throughput, control overhead and robustness [11]

To organize a multicast overlay, a natural structure is a tree, which, however, is known to be vulnerable to end-hosts dynamics. Data-driven approaches address this problem by employing a mesh structure, which enables data exchanges among multiple neighbors, and thus, greatly improve the overlay resilience. This hybrid design, referred to as the mTreebone, brings a series of unique and critical design challenges. In particular, the identification of stable nodes and seamless data delivery are difficult in the mTreebone. Almost all Multimedia applications involve video streaming over an overlay network. In the overlay networks, throughput, packet loss rate, and delay suffer mainly, when there is an event of a random node join/leave in the network.

A buffer is a temporary location to store the data in hardware or software. Buffers are used whenever the inflow data is faster than the process or outflow data. Buffers should be optimized in size in order to work efficiently. However, sometimes buffers, just like any other storage device, can become too full of data. For example, if a buffer receives too much data, and cannot process it fast enough, then some of the received data may be dropped by the buffer. Similarly, the buffers contained in the routers may also become full, as one of the networks, the router, is connected to become congested. As a result, the router and buffer may drop packets of data.

In order to reduce the probability of dropping packets of data, larger buffers at each router and/or multiple buffers at each router may be employed. While the larger buffers will certainly minimize the packet loss in the network in case of traffic congestion, as the buffers fill up, the delays encountered by each packet would increase dramatically. On the other hand, if the buffers are too small, while the delay may be reduced, the packets will be dropped too frequently. Hence, there is a trade-off between a buffer that is small enough to meet the packet delay requirements and, at the same time, large enough to have a high throughput. So there is a need to find out an optimal buffer threshold value. In general, to improve the throughput and packet loss rate, the network coding technique is used [17].

Network coding is a technique which can be used to improve a network's throughput, efficiency and scalability, as well as resilience to attacks and eavesdropping. Instead of simply relaying the packets of information they receive, the nodes of a network take several packets and combine them together for transmission. This can be used to attain the maximum possible information flow in a network. It has been proven that linear coding is enough to achieve the upper bound in multicast problems, with one or more sources. Random network coding is a simple yet powerful encoding scheme, which in broadcast transmission schemes allows close to optimal throughput using a decentralized algorithm. Nodes transmit random linear combinations of the packets they receive, with coefficients chosen from a Galois field. If the field size is sufficiently large, the probability that the receiver(s) will obtain linearly independent combinations (and therefore obtain innovative information) approaches one [19].

This research work proposes a novel Push/Pull Smooth video Streaming Multicast (PPSSM) protocol, to improve the network throughput and packet loss rate, by using an efficient organization of stable nodes in the tree-mesh topology, combining network coding with the buffer management technique. As the proposed protocol handles the video packets distribution in a more efficient way than the other ALM protocols, it can be used in applications like multiplayer online gaming, e-learning, multi-party video conference, tele-medicine, etc. The proposed protocol involves the following three important stages, to achieve improved throughput and minimized packet loss rate.

1. Formation of a tree-mesh overlay structure using more stable nodes.
2. Efficient buffer management using techniques, such as IPA and Stochastic Approximation.
3. Network Coding.

Section 2 of this paper will discuss the literature survey related to the proposed PPSSM protocol; section 3 will discuss the various stages of the development of the proposed protocol, viz., tree-mesh topology construction, buffer management and network coding. The performance analysis of the proposed protocol and its comparison with the existing protocols, such as CoolStreaming and mTreebone, will be discussed in section 4. Section 5 concludes with the summary of the proposed protocol.

2 Related works

This section discusses the various existing protocols, which used different techniques for the distribution of video packets, related to the proposed PPSSM protocol.

CoopNet by Padmanabhan, V. et al. [15] provides a resilient technique to deliver streaming contents from a single source. It makes use of multiple descriptions coding

(MDC) and multiple distribution trees to achieve robust data delivery. MDC is a method of encoding audio and/or video signals into M > 1 separate streams, or descriptions, such that any subset of these descriptions can be received and decoded into a signal with distortion (with respect to the original signal), depending on the number of descriptions received: the more descriptions received, the better the quality of the reconstructed signal. In CoopNet, each sub-stream is delivered using a different distribution tree (formed by the same set of members). This delivery mechanism provides robustness, as the probability of all streams concurrently to fail to arrive is very low, when M is sufficiently large.

Split stream [4] is a single-source, multiple-receiver, multi-tree overlay utilizing a push-based approach in which the source disseminates data over several interior-node disjoint trees. Since the root and all the other interior nodes will, if possible, be different for every tree, the bandwidth cost of relaying data is distributed among all participants.

Chunkyspread [21] is an unstructured P2P multicast protocol; it is simple and robust. It uses multiple trees to provide fine-grained control over member load, reacts quickly to membership changes, scales well, and has a low overhead. It exhibits far better control over transmit load than Split stream, while exhibiting comparable or better latency and responsiveness to churn. This comparison establishes Chunkyspread as the 'best of breed' among treebasedP2P multicast algorithms.

Banerjee et al. [2] discussed the NICE protocol (Internet Co-operative Environment) which is a hierarchical tree based protocol for multicast routing. They constructed the hierarchical structure of the overlay nodes using metrics related to delay and bandwidth, for low latency distribution. Using NICE, it is possible to control the variations in the hierarchical multicast trees with a low control overhead.

Zigzag byTran et al. [20] is a single source, degree-bound application layer multicasting approach for media streaming. It organizes the receivers into a hierarchy of clusters and builds a multicast tree on top of it. It consists of two important entities: 1) the administrative organization representing the logical relationships among the peers, and 2) the multicast tree representing the physical relationships among them. The join request propagates down the multicast tree, until a suitable parent is found. It finds a node that is closest to the lowest layer. The ZIGZAG periodically runs optimization algorithms to improve the quality of service to the clients. Degree based and capacity-based switching approaches are taken to balance the degree and load of the nodes respectively.

BiTos is proposed by Aggelos Vlavianos [23]; it provides the video in order, enabling smooth playback, and the rarest first order, enabling the use of parallel downloading of pieces. The author introduces three different piece selection mechanisms and they evaluate them through simulations based on how well they deliver streaming services to the peers [22].

YuningSong [18, 13], formulatedthe minimal exposure problem (MEP) and the related models, and then converted MEP into a Steiner problem, by discretizing the monitoring field to a large-scale weighted grid. Inspired by the path-finding capability of physarum, he developed a biological optimization solution to find the minimal exposure road-network among multiple points of interest, and presented the physarum optimization algorithm.

Besharati et al. [3] described a binning technique to cluster nearby receivers based on the distance from a constant number of landmarks. Landmarks are the distribution centres in the overlay network. Then a k-array tree was constructed over the clusters, with the most stable nodes in each cluster as the head. The heads decide where it is necessary to split/merge the clusters, when a node joins/leaves. They used a metric based on RTT to measure the weighted distance from landmarks.

CoolStreaming/DONet: Zhang et al. [26] described a push/pull mechanism for data exchange in overlay networks. They used random network coding with random gossiping for disseminating video blocks in a peer-to-peer live streaming network. Every node periodically exchanges data availability information with a randomly chosen set of neighbors, and retrieves data unavailable from one or more neighbors, or supplies data to the neighbors. They provided experimental results which prove the robustness and scalability of the ALM protocol. Though the CoolStreaming protocol used the push/pull technique, the lack of buffer management caused low network performance, throughput and high packet loss.

mTreebone: Wang et al. [24] discussed the mTreebone protocol, which combines the advantages of both the tree and the mesh overlays. The network performance depended upon a stable subset of nodes known as the backbone. The treebone is a tree layout of the backbone nodes in the network. The non-stable nodes appear on the outskirts of the treebone as leaves. The stable nodes along with other nodes are organized using an auxiliary mesh overlay, which provides resilience to handle peer dynamics and exploit the available bandwidth. The authors provided algorithms for the construction and control of the treebone structure. Though the mTreebone protocol is constructed using stable and unstable nodes, they did not use any efficient buffer management and network coding for the distribution of video packets.

To optimize the network traffic in a wireless mesh network, Hongju Chenga [5] proposes the channel assignment algorithm named as Discrete Particle Swam Optimization algorithm for the Channel Allocation (DPSO-CA) problem by introducing the genetic operations into the iteration process. The Coolest Path Protocol [25] is used to calculate the maximum stability cost over the path links or a mixed cost between the maximum and the accumulated cost along the path.

In today's IP networks, there are three primary methods for setting the buffer sizes. However, these methods were developed in order to minimize the packet loss for TCP-based large file transfer applications. As discussed earlier, the TCP is not suitable for interactive delay-sensitive applications, e.g. real-time. The three methods are: Bit rate-Delay Product (BDP), Over-Square Root, and Connection Proportion Allocation. All these three methods used the Round Trip Time (RTT) to set an optimal buffer size, where as in the UDP datagram streaming applications, the RTT is not used to set a buffer size [12]. So an alternative technique has been proposed by Markou et al., [14] who presented a dynamic control of buffer size in wireless networks. The parameter considered is loss and workload costs, which are constantly monitored. The buffer threshold is then set according to the cost at various times. They propose the use of the Infinitesimal Perturbation Analysis (IPA) algorithm for cost monitoring, and the Stochastic Approximation algorithm for setting the threshold value accordingly. The simulation results indicated that this technique exhibits good convergence properties.

When number of participants increases, Quality of Service (QoS) is likely to deteriorate [27]. Hence, to provide good Quality of Service (QoS) in the peer-to-peer network, which replaces content delivery network for scaling number of participants, distributed algorithms [29] have been introduced. To treat the QoS in the resource limited heterogeneous network, Liang Zhou [28] developed, a joint forensics-scheduling scheme, which allocates the available network resources based on an affordable forensics overhead and expected QoS.

Ho and Lun [10] proposed a distributed random linear network coding approach for multi-source multicast networks, and showed the advantages of randomized network coding over routing in such scenarios. They provided the success probability of such a coding scheme in arbitrary networks, and generalized the method to work with correlated sources. It was also used for reliable multicast [16, 8] for wireless networks.

The following section will discuss the proposed Push/Pull Smooth video Streaming protocol to enhance throughput, by integrating the merits of both the CoolStreaming and mTreebone protocols.

3 Push/pull smooth video streaming protocol (ppssm)

This section proposes a novel Push/Pull Smooth video Streaming protocol, which uses the tree-mesh overlay structure formation, dynamic buffer management, and network coding technique, to improve the performance metrics such as throughput, packet loss rate and control overhead.

Figure 1 shows the overview of the proposed PPSSM protocol. According to the proposed protocol, initially, the source node will split the incoming MPEG4 encoded video into a number of packets using the input buffer, and send them through the multicast tree-mesh topology constructed, using the overlay nodes in the network. Once the tree-mesh topology is

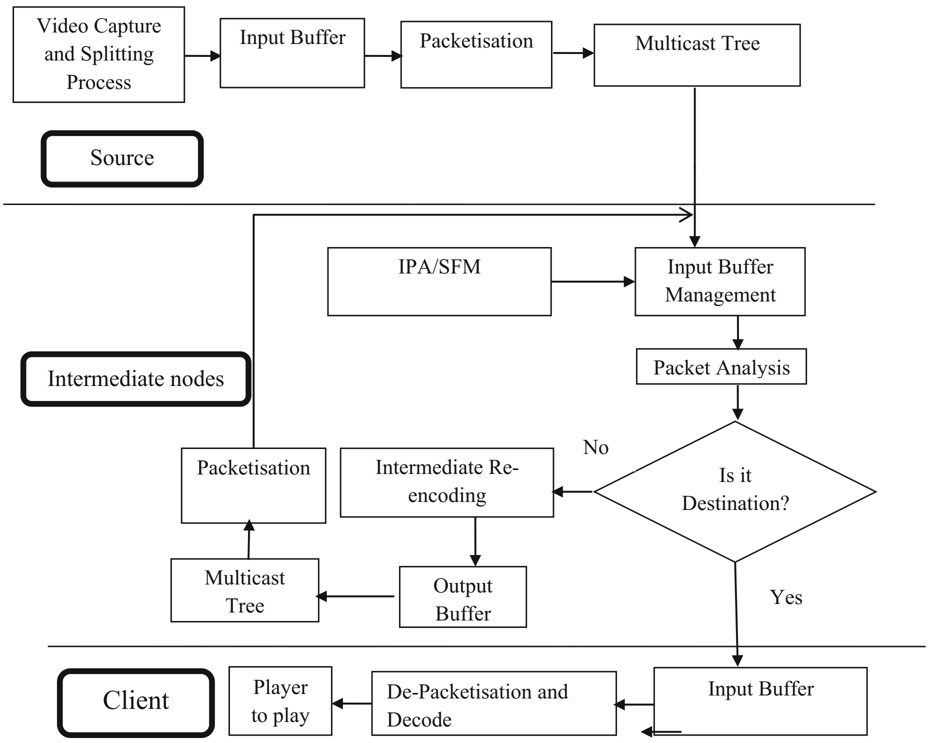


Fig. 1 Push/Pull Smooth video Streaming Protocol Architecture

formed, based on the capability of the nodes, dynamic buffer management is initiated to minimize the packet drops. Once the packets are queued in the buffer, the network encoding will be performed and the queued packets forwarded to the destinations. If the forwarded packets are received by the intermediate nodes which are not the destination, then they will be again network coded and forwarded to their children nodes. As and when the coded packets reach the destination, i.e., the client, the network decoding process will be commenced, to play the video streaming.

The effective construction of a hybrid tree-mesh topology is an important stage of the proposed protocol, because this determines the length of the tree, which is responsible for the packet drops and link failure. If the length of the tree is increased the probability of link failure is also increased, which in turn, affects the packet distribution. The tree is constructed, based on the stable nodes, and the mesh is constructed using unstable nodes. Also in the tree, the parent-children nodes are organized based on the degrees of the related nodes. If a node is stable and has a higher node degree, then it will be kept as a parent. Meshes are constructed using unstable nodes, and they will support the trees. In video streaming, the streams are relayed using the intermediate nodes present in the tree and mesh. During transmission in the tree, if any data block is missing, then the missing data block is pulled from the nodes in the mesh. Effective buffer management will minimize the data block loss. This buffer management is achieved by setting an optimal threshold value, at every time instant dynamically as the video packets arrive, and as the traffic conditions change. Then, using the Infinitesimal Perturbation Analysis algorithm and the Stochastic Approximation Algorithm, an optimal buffer threshold value is set.The network coding is performed on the data blocks, which are from the incoming video streams. An efficient tree-mesh construction and management, dynamic buffer management, and network coding are the three important phases of the Push Pull Smooth video Streaming protocol.

The following subsections will discuss the tree-mesh construction and maintenance, network coding and dynamic buffer management.

3.1 ALM tree-mesh construction and management

This tree-mesh construction and management is the main phase of the PPSSM, because all the video streaming blocks are forwarded through these tree-mesh overlay nodes. Hence, it is important to construct and manage an efficient tree-mesh to realize a live streaming. The ALM tree-mesh construction depends mainly on the detection of stable nodes. After identifying the stable nodes, the tree is constructed using the node degree. The node degree depends on the uplink bandwidth of the related node. If the bandwidth is high, then the node degree of the related nodes is also high. This Tree-mesh is reconstructed as and when any node is entered in the tree mesh, or any node in the tree-mesh leaves it. Hence, tree management is initiated on a new node's arrival, or an existing node's exit from the tree-mesh. The following sub section will discuss the detection of stable nodes, tree-mesh construction and tree-mesh management.

3.1.1 Detection of stable nodes

In this tree mesh construction, the tree consists of stable nodes, and the mesh consists of unstable nodes. The proposed PPSSM optimizes the stable nodes in the tree which minimizes or eliminates the pull operations from the unstable mesh overlay nodes, by exploring the potential of the stable nodes. The stable nodes are identified, using their Expected Service

Time (EST). This EST represent show long the node will live in the network to provide service. Hence, the age of the node depends on the EST. If the age of the node is greater, then the corresponding node is identified as a more stable node when compared to others. Identifying those more stable nodes is done by fixing the threshold level in EST. Improving the identification of more stable states is achieved by maximizing the EST value. This EST value is estimated using Equation (1).

$$\mathrm{EST(t)} = \ (\mathrm{L}-\mathrm{t}) \left(\frac{k\left(\frac{1}{x_t^{k-1}} - 1\right)}{(k-1)\left(\frac{1}{x_t^k} - 1\right)} - 1 \right) \tag{1}$$

where L is session length, L-t is residual session length at time ‘t’, EST(t) expected service time in tree bone for a node arriving at ‘t’, Xt is normalized age threshold and ‘k’ is the shape parameters of pareto distribution. The effectiveness of the tree depends on the age threshold. If the threshold is too low, many unstable nodes would be included in the tree. Similarly, a high threshold minimizes the number of stable nodes present in the tree. Hence, optimizing the EST is important for selecting the optimal number of stable nodes in the tree. The optimal number of stable nodes pushes the video stream at the maximum level to the destination, and minimizes the pull operation from the mesh overlay nodes. Once a stable node is in the tree, it remains there until it leaves or the session ends.

3.1.2 Tree–mesh topology construction

A tree-mesh network is constructed, based on the degree of the nodes. The *Node Degree* represents the maximum number of child nodes that supports a particular node. The higher degree nodes are placed at the top of the tree. The degree of the node depends upon its uplink bandwidth. When the bandwidth of the node is increased, then the degree of the node is also increased. The degree of the node is calculated by using Equation (2).

$$d_n(T) = \mathrm{b(n)/r} \tag{2}$$

where d_n (T) denotes the degree of the node n of the multicast tree T. d_n (T) implies that node n can simultaneously forward the incoming packets to at the most d_n $(T) - 1$ other nodes. ‘r assumes that the transmission rate is a constant bit rate, and$b(n)$ is the available bandwidth. The node corresponding to the maximum degree in a multicast tree will act as the parent, and the other nodes in the multicast tree, as the children. An example of the tree-mesh topology overlay network is shown in Fig. 2. In the Fig. 2, S is the source and root node having the highest degree; T_1, T_2,…T_8 are the stable tree nodes and M_1,M_2,…M_{32} are the unstable mesh nodes. The nodes in the tree are arranged, such that the following criterion holds good.

$$d_s(T) \leq d_{T1}(T) \leq d_{T2}(T) \leq \ldots \leq d_{Tn}(T) \tag{3}$$

3.1.3 Tree construction

Figure 3 shows the tree topology after arranging the nodes, based on their degree. In this Fig. 3, the higher degree nodes will act as the grand parent and parent, and the lower degree

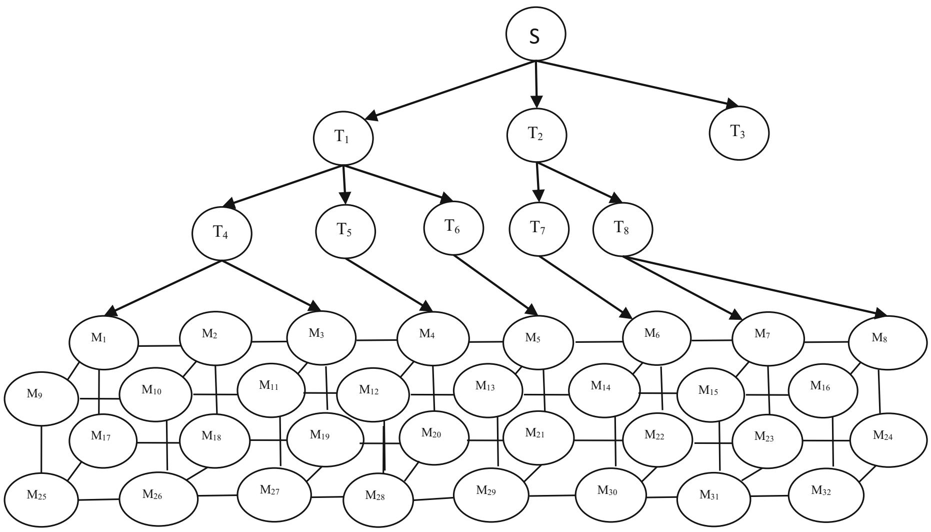
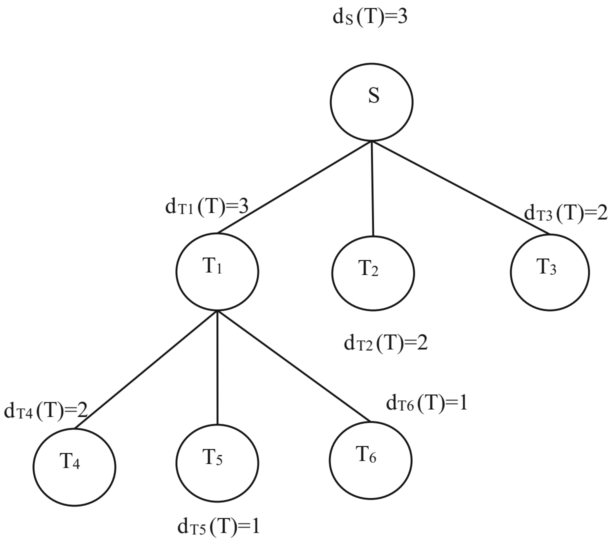


Fig. 2 Tree-Mesh topology

node will act as the child. Referring to 3, the source node ‘S’ has a maximum degree of 3. Hence, node ‘S’ gets attached with three children nodes. These attached three children nodes also have comparatively higher degrees only. In this example, the source node ‘S’ has three children ‘T_1’, ‘T_2’, and ‘T_3’ and their node degrees are 3, 2 and 2 respectively. Now among the ‘T_1’, ‘T_2’, and ‘T_3’ nodes, node ‘T_1’ has more degrees compared to the other two. If the number of the remaining nodes is less than or equal to 3, then those remaining nodes will be attached to node ‘T_1’.Otherwise, the first three higher degree remaining nodes will be attached to node ‘T_1’ because its node degree is 3, and the remaining lower degree nodes will be attached according to the degrees of node ‘T_2’ and node ‘T_3’. Algorithm 1 shows that, how the multicast tree is being constructed.

Fig. 3 Tree Construction



```
Algorithm 1: FormTree(List li)
set newlist ← li
set Number m:=1
for i:=0 toli_size
do
assign temp_node:=li.get(i)
if m < newlist_size
then
for j:=0 to temp_node_degree
do
if children[j] is null
then
children[j] ← newlist.get(m)
    children[j]_parent ← temp_node_name
  m ← m + 1
else
break
```

3.1.4 Mesh construction

Nodes which do not satisfy the threshold EST(t) value are identified as unstable nodes, and those unstable nodes are fully connected with each other, forming a mesh. The reason for forming the fully connected mesh is to ensure the data availability among the unstable nodes. As the unstable nodes have a low age in the network, the data may be lost if they are connected in a tree structure. Hence, to ensure the data availability, and as and when any data block is lost in the tree structure, the tree will pull the related data block from the nodes in the mesh. The data for the mesh will not only be received from the single subtree, but also from all other possible sub-trees. If a subtree misses a data block, then the corresponding data block may be retrieved, using the nodes in the mesh network.

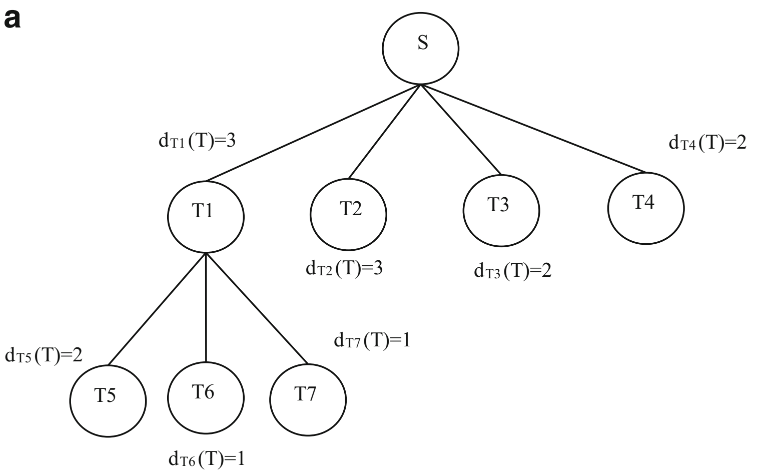
3.1.5 Tree-mesh management

In this PPSSM protocol, tree management has an important role in forming an efficient tree based on the degree of the node. Tree management and reconstruction will be initiated when, i) a node enters the tree, ii) leaves the tree iii) or it is pruned.

Case 1: Node join In a network, many trees are constructed according to the application. If any node is interested to join a particular application group tree, first it approaches the server. The server will give the details about the closest root of the particular application group tree to the requested node. If the node gets the details about the closest root node of the tree, then it will send the join request to the root of the particular tree. As and when the join request of the node reaches the root node, the reconstruction of the tree will be initiated. First, the degree of the requested node will be compared with the degree of the root node. If the degree of the root node is lesser than that of the requested node, then the requested node will act as a root, and the root node will join as a child to the requested node, which is now the root.

This reconstruction process continues till it reaches the correct position of each and every node in the tree. Finally, the tree will be reconstructed according to the degree of the nodes in the tree. Consider the example tree shown in Fig. 3; if a node having degree 4 joins the tree, then Fig. 4a shows the reconstructed tree when the requested node's degree is 4 after joining the tree. In Fig. 4a, the root node 'S' is the newly joined requested node, having node degree 4. The existing root node 'S' will now act as the tree node T_1. Similarly, Fig. 4b shows the reconstructed tree when the requested node's degree is 5.

If the degree of the requested node is less than the degree of the root node, then the requested node will check the degrees of the child nodes of the root node. In case the first child



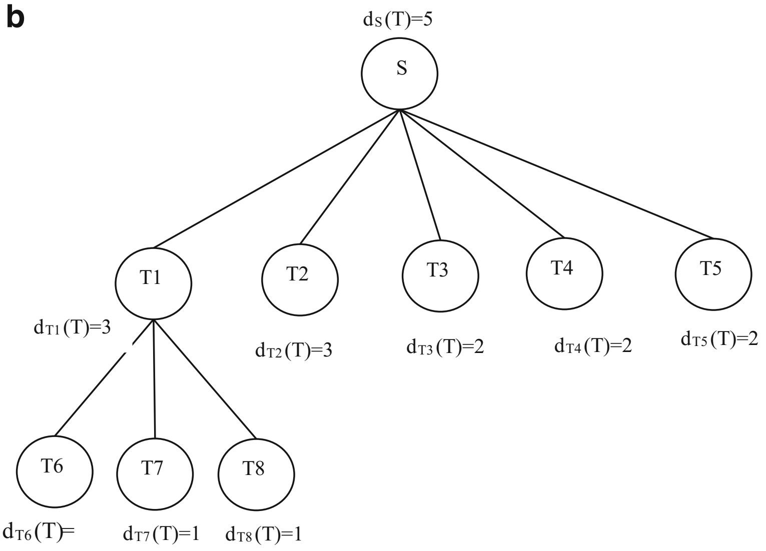


Fig. 4 a Node join with degree 4. b Node join with degree 5

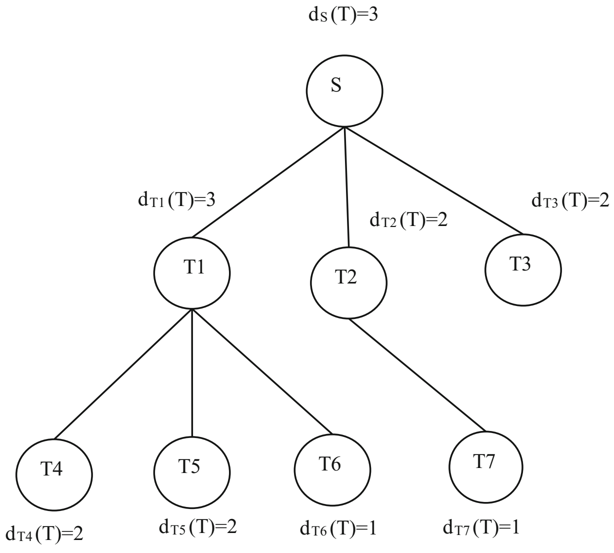


Fig. 5 The requested node's degree is less than the root's degree

node's degree is greater than the requested node's degree, then the requested node is compared with the remaining child nodes' degrees. If the entered node's degree is equal to the child node's degree, then it will move and compare with the degree of the next child node. This process continues till it reaches the correct position in the tree, based on the ascending order of the degree of the nodes. Figure 5 shows a reconstructed tree when the requested node's degree is less than the root's degree of the example tree shown in Fig. 3; i.e., the requested node's degree is 2 and the root node's degree is 3. Algorithm 2 shows that, how the new node is joined to a multicast tree.

```
Algorithm 2: Node_Join(Node n, Number degree).
for i:=0 to degree
do
children[i] ← null
end
list ← add node n
sort(list) based on degree
FormTree(list)
```

Case 2: Node Leave If any node in the tree is willing to leave, then the tree reconstruction process will be initiated. The node leave request will be sent to its parent, if any node is leaving the tree. Then that leave request will be propagated to its root. As and when the root node receives the leave request, it will send back the leave acceptance response to the requested node. After the receipt of the leave acceptance response, the requested node will quit the tree. Then the remaining nodes will be reorganized according to their nodes' degrees. In the example tree shown in Fig. 3, node 'T_1' is willing to leave the tree; then, it sends the leave request to its parent, which is the root of the tree. Now, the root will send back the leave acceptance to node 'T_1'. Figure 6 shows the

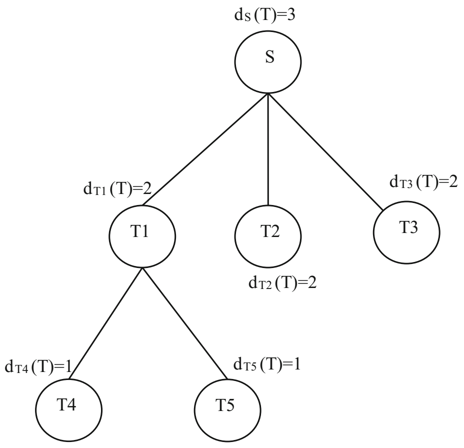


Fig. 6 After the deletion of a node 'T1' from tree

reconstructed tree after the exit of node 'T1' from the tree. Algorithm 3 shows that, how the node, left of the multicast tree.

```
Algorithm 3: Node_Leave(List li, Node n).
if li is empty
then
        break;
else
        set key:=node_name
        if li.get(key) is not null
        then
        li.remove(key)
sort(list) on based degree
FormTree(list)
```

Case 3: Prune In general, all the parent nodes are continuously checking the aliveness of their children nodes, by using hello packets. If a particular node is continuously not acknowledging the hello packets sent by its parent, then the related node is identified as a not alive node. Then this not alive node will be removed from the tree by its parent. The deletion process of the node and reconstruction of the tree will be similar to the node leave process.

3.2 Buffer management

Effective resource management is important in networks, which have heavy and unpredictable traffic. The major concern in video data blocks streaming is an efficient management of the buffers at the intermediate nodes. For buffer management, the basic control parameter called the buffer threshold (θ) has to be dynamically set, at every time instant the video packets arrive, and as traffic conditions change. If θ is small, the probability of the buffer overflow is high and the workload or packet delay is less. As θ increases, the packet loss probability drops, but the workload increases. In order to modify the buffer threshold, some performance measure should be continuously monitored. The Infinitesimal Perturbation Analysis (IPA) algorithm is used to obtain the sensitivity estimates of the various costs with respect to the buffer threshold. These estimates are used in the Stochastic Approximation (SA) algorithm, to set the optimal buffer threshold value.

The objective of buffer management is to minimize the packet loss and packet delay/ workload. All these packet losses and packet workload depend on the buffer threshold. Hence, the objective cost function $E[J]$, in terms of the packet loss cost parameter and packet delay/ workload is set, using Equation (4). This objective function is to be minimized, and it will always be monitored.

If Q (θ) is the cost due to the packet workload/packet delays, L (θ) the cost due to packet loss, and RC the rejection cost of each packet in the buffer; then, $E[J]$ represents a trade-off between providing low delay and low packet loss probability, and it is estimated using Equation (4).

$$\min_{\theta} E[J] = Q(\theta) + RC.L(\theta) \tag{4}$$

Where J is the total cost that should be minimized. If the buffer threshold 'θ' is controlled, then the objective function J is also controlled. Hence, the optimization of 'θ' will considerably reduce the packet loss and packet delay.

From Equation (4), it is understood that J is directly proportional to packet delay, and packet loss, and rejection cost. Hence, if J increases then the packet loss and packet delay will also increase. Similarly, if J decreases it will decrease the packet loss and packet delay. An increase in J requires the expansion of the buffer to minimize the packet loss. The minimized J will show that the buffer size is more than sufficient, and hence, the buffer size may be reduced. The minimization of J is achieved by using Equation (5).

$$\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta} = \frac{\mathrm{dQ}(\theta)}{d\theta} + RC.\frac{\mathrm{dL}(\theta)}{d\theta} \tag{5}$$

To optimize 'θ' it is important to optimize J. As $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$ depends on $\frac{\mathrm{dQ}(\theta)}{d\theta}$ and $\frac{\mathrm{dL}(\theta)}{d\theta}$, these sensitivity estimates $\frac{\mathrm{dL}(\theta)}{d\theta}$ and $\frac{\mathrm{dQ}(\theta)}{d\theta}$are obtained through Algorithm 4 (IPA).

Algorithm 4: Infinitesimal Perturbation Algorithm.
```
initialized a packet loss counter $L = 0$ and a cumulative timer for
work load/delay $W = 0$
i ← 0
if an overflow event is observed at time t and $i = 0$ then
        i = t
end
if a busy period ends at time t and $i = 0$ then
        L ← L + 1
        W ← W + (t – i)
        i ← 0
        if t = T and $i = 0$ then
                L ← L + 1
                W ← W + (t – i)
        end
end
```

The final values of *L* and *W* divided by the length of the observation interval *T*, provide the IPA derivative estimates $\frac{\mathrm{dL}(\theta)}{d\theta}$ and $\frac{\mathrm{dQ}(\theta)}{d\theta}$ respectively. Substituting estimates $\frac{\mathrm{dL}(\theta)}{d\theta}=\frac{\mathrm{L}}{T}$ and $\frac{\mathrm{dQ}(\theta)}{d\theta}=\frac{\mathrm{W}}{T}$which are derived using IPA in Equation (5), the estimated value of $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$ is obtained using Equation (6).

$$\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}=\frac{\mathrm{W}}{T}+RC.\frac{\mathrm{L}}{T} \tag{6}$$

To optimize the estimated objective function $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$, the Algorithm 5 (SA) is used.

Algorithm 5: Stochastic Approximation Algorithm.

```
Initialization: Set an initial, arbitrary value $_0$ for the buffer
size.
for each step n do
    Calculate the estimate $\frac{\mathrm{dJ}_{\mathrm{T}}(\theta)}{d\theta}$;
    if $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$= 0 then
      set $_{n+1}$ = $_n$ – s. $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$
    else
        compute appropriate $\frac{\mathrm{W}}{T}$ and $\frac{\mathrm{L}}{T}$ values using IPA to optimize $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{d\theta}$
    end
end
```

If θ0 for the buffer size is a very high value, it is possible that no packets are lost in the interval [0,T] and thus $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{\mathrm{d}\theta}$ will evaluate to 0 (according to the IPA algorithm). However, this does not necessarily imply an optimal buffer size (due to excessive workload); therefore, when $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{\mathrm{d}\theta}$ evaluates to 0, the θ is reduced by s. $\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{\mathrm{d}\theta}$, where 's' is the step size. If$\frac{\mathrm{d}_{\mathrm{J}}(\theta)}{\mathrm{d}\theta}$ does not reach zero, then invoke IPA to compute the appropriate $\frac{\mathrm{W}}{\mathrm{T}}$ and $\frac{\mathrm{L}}{\mathrm{T}}$ values to optimise'θ'.

Thus, the optimization of the objective function J is achieved by dynamically controlling the buffer threshold 'θ'. The optimal buffer threshold minimizes the packet loss and increases the throughput considerably.

3.3 Network coding

Encoding is done by splitting the input packets into a suitable size, and converting them into the matrix format. This is then combined with the finite field to produce the encoded matrix. The encoding factor is added into the header information of the encoded data, which is to be forwarded. This encoding factor is used in recovering the original data at the receiver side when decoding. This network coding process increases the throughput and minimizes the end-to-end delay. Hence, the proposed scheme uses network coding to improve the quality of video streaming.

In the proposed scheme, each 'K' number of packets in the source node are network coded and forwarded to the next intermediate nodes. The ssource node has K message packets, such as $\mathrm{w}_1,\mathrm{w}_2\ldots\mathrm{w}_\mathrm{K}$, which are vectors of length λ over some finite field F_q. (If the packet length is b bits, then we take λ = ┌b/log(2q)┐ .) The coding operation is performed by the source node, as well as every intermediate node. The receiver nodes will perform the decoding operation.

The coefficients of the combination are drawn uniformly from F_q. Since all coding is linear, any packet u_i in the network as a linear combination of $w_1, w_2 \ldots w_K$, namely, $u_i = \sum_{k=1}^{K} \gamma_k w_k$, $i = 1,2.m$, where γ is the global encoding vector of u_i, and it must be sent along with u_i, as side information in its header.

Now, an intermediate node R receives 'm' number of network coded information from the source node. The received network coded packets are u_1, u_2, $u_3 \ldots, u_m$. Using these received network coded packets and the global encoding factors $\beta_1, \beta_2, \beta_3 \ldots \beta_m$, the intermediate node R will initiate the network coding operation. Now R will recreate 'l' new encoding information data v_1, v_2, $v_3 \ldots$, v_l, using Equation (7), and it will forward it to the next set of intermediate nodes.

$$v_j = {}^{m}\sum_{i=1} u_i \gamma_i \quad j = 1, 2, .., l \tag{7}$$

If the received node is not a leaf node then this network coding process continues till it reaches a leaf node. All intermediate nodes forward the packets to their children. The decoding operation will be initiated at all the intermediate and leaf nodes of the tree, which participated in the session. If the receiver child node receives 'n' encoding data $E_1, E_2, \ldots E_n$, and the global encoding factors are $\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n$, then it recovers its parent data using the following decoding Equation (8).

$$A_i = {}^{n}\sum_{j=1} E_j \alpha_j^{-1} \quad i = 1, 2 \ldots p \tag{8}$$

Similarly, the destination node recovers 'k' original information data packets $w_1, w_2 \ldots w_K$, which were multicast by the source node through the following decoding Equation (9).This decoding operation uses the inverse of the global encoding factors for u_i(i = 1, 2, 3…,m).

$$W_i = {}^{m}\sum_{j=1} u_j \gamma_j^{-1} \quad i = 1, 2 \ldots k \tag{9}$$

On the whole, section 3, elaborated the important stages, such as tree-mesh construction, dynamic buffer management, and network coding techniques, of the proposed PPSSM protocol. The following section will discuss the performance evaluation of the proposed protocol.

4 Performance evaluation

This section will evaluate the performance of the proposed PPSSM protocol, in respect of varying the total number of overlay nodes in the tree-mesh topology, buffer size, and the number of receiver nodes. We have conducted extensive simulations and PlanetLab based experiments to evaluate PPSSM. PlanetLab provides a research platform for large scale distributed experimentation of peer-to-peer systems over the Internet. We randomly selected nodes for different experiments to validate our results. Each experiment was repeated ten times and we show the confidence intervals. The source sends streaming bit rates of 400 kbps to 500 kbps. The source was located on a host at Anna University. The source will send about 30 frames per seconds. We used the TCP protocol for transmission. We used a maximum of 2048 nodes in our experiments. The source sends one chunk of the data stream down to the tree each second. Each node forwards that chunk onto each of its children as soon as possible. Each node is configured, and it managed the children based on their uplink bandwidth speed. The session length is set to 3600 s, by considering a one hour teaching session, and each data block is a 1-s video; there are 5000 overlay nodes considered for forming the overlay network. The maximum end-to-end delay is 1000 ms between two overlay nodes, and the maximum upload bandwidth is uniformly distributed from 512 kbps to 1Mbps bandwidth required for a full streaming. For comparison, the total number of nodes in the tree-mesh topology is varied from 8 to 2048. The default age threshold is set to 30 % of the residual session length.

The performance metrics evaluated are packet loss, average decoding time, control overhead and throughput. Finally, the performance of the proposed protocol is compared with those of the existing CoolStream and mTreebone protocols. The definitions of the performance parameters and metrics are discussed in the following sub sections.

4.1 Performance parameters

i. Total number of overlay nodes in the tree-mesh topology: this is the total number of overlay nodes involved in constructing a tree-mesh topology for video streaming.
ii. Buffer size: A buffer is a temporary location to store or group information in hardware or software. Buffers are used whenever data is received in sizes or at rates that may be different from the ideal size or rate.
iii. Number of receiver nodes: this is the total number of receiver nodes in the multicast group of the tree-mesh topology. The maximum number of receiver nodes will be the total number of overlay nodes in the tree-mesh topology other than the source node.

4.2 Result and analysis

This section analyses the performance of the proposed PPSSM protocol and compares it with those of the existing CoolStreaming and mTreebone protocols.

4.2.1 Number of nodes Vs packet loss

Figure 7 shows the graph of the number of nodes and the packet loss rate between the proposed PPSSM protocol, and the existing mTreebone and Coolstreaming protocols. When the number of nodes is increased, the packet loss rate also increased due to link failure between the overlay

Fig. 7 Number of nodes Vs Packet loss

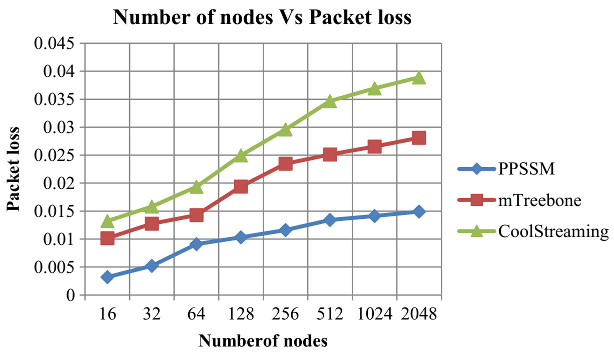
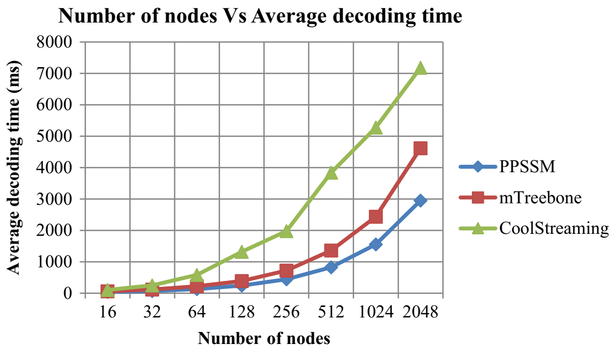


Fig. 8 Number of nodes Vs Average decoding time

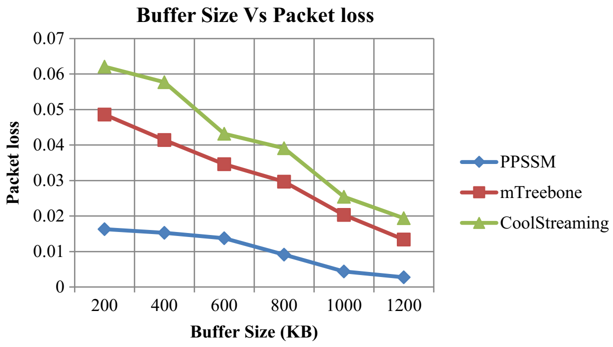


nodes, congestion in the network, processing delay etc. When compared to the mTreebone and Coolstream, the proposed PPSSM shows improved performance in minimizing the packet loss rate, due to the effective management of the buffer, and organization of the overlay nodes based on their capacity. If the minimal number of nodes formed the tree-mesh topology then the packet loss rate is very low, viz., about 0.32 %, which is 68.53 % and 75.79 % less, compared to the existing mTreebone and Coolstreaming protocols respectively. Similarly, if the number of nodes is increased to 2048 in forming the tree-mesh topology, then also the proposed PPSSM protocol shows 46.96 % and 61.69 % improvement over the other two existing protocols.

4.2.2 Number of nodes Vs average decoding time

The average decoding time of the proposed PPSSM protocol is compared with that of the existing mTreebone and Coolstreaming protocols, and shown in Fig. 8. When the number of nodes is increased, according to the proposed PPSSM protocol, the tree length is increased based on the node degree of the overlay nodes. If the node degree is increased, then the tree length is decreased. The decrease in tree length will minimize the network encoding, which is directly proportional to the network decoding time. From the graph shown in Fig. 8, it is inferred that as the proposed PPSSM protocol effectively reduces the tree length compared to the other existing two protocols, the average decoding time of the proposed protocol is considerably reduced over the others. As 2048 nodes are involved in forming the tree-mesh topology, to transfer the video stream the proposed PPSSM protocol takes 2947 ms on an average for decoding. But for the same scenario, the existing mTreebone and CoolStreaming

Fig. 9 Buffer Size Vs Packet loss



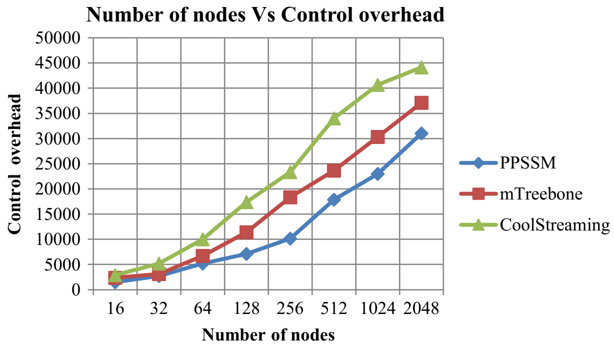


Fig. 10 Number of nodes Vs Control overhead

protocols take 4613 and 7181 ms for decoding. The proposed protocol improves the average decoding time by35.5 % to 42.75 %, and 58.97 % to 81.8 %, over the existing mTreebone and CoolStreaming protocols respectively.

4.2.3 Buffer size Vs packet loss

Packet loss in the network is directly proportional to the buffer size of the nodes. As the proposed PPSSM protocol effectively manages the buffers of the nodes, the packet loss is considerably reduced compared to the other existing protocols. Figure 9 shows the graph of the buffer size and the packet loss rate, between the proposed PPSSM protocol and the existing mTreebone and CoolStreaming protocols. When the buffer size is small, as the existing protocols did not have any buffer management mechanism, they easily drop the packets when buffer overflow occurs. But the proposed PPSSM manages the buffer size of the nodes efficiently, according to the incoming packets, and the packet drop level is reduced over the other protocols. When the buffer size is increased from 200 to 1200 KB, the packet drop rate of the proposed protocol is decreased from 1.63 % to 0.27 %. For the same scenario, the existing mTreebone and Coolstreaming protocols decrease the packet drop from 4.86 % to 1.34 % and 6.21 % to 1.94 % respectively.

4.2.4 Number of nodes Vs control overhead

In general, when varying the number of nodes due to the packet drops, link failures, and node leave/join requests, the control overhead in the network is increased. Figure 10 compares the

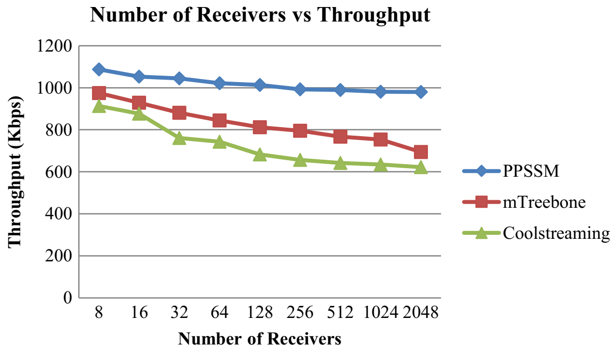


Fig. 11 Number of Receivers Vs Throughput

control overhead of the proposed PPSSM and the existing mTreebone and Coolstreaming protocols. As the proposed PPSSM protocol minimizes the packet drop rate and tree height, the transmission of the control packets in the network is considerably minimized. From the results it is inferred, that the proposed PPSSM protocol minimizes the control overhead on an average by 27.27 % and 47.53 % respectively, over the existing mTreebone and Coolstreaming protocols.

4.2.5 Number of receivers Vs throughput

The comparison of the number of receivers and the throughput of the proposed PPSSM protocol and the existing mTreebone and CoolStreaming protocols is illustrated in Fig. 11. When the number of receivers is increased, naturally, the throughput of the network is reduced considerably, due to network congestion as well as node failure. As the proposed PPSSM protocol uses network coding, the throughput is increased, compared to the other existing protocols. Also, the efficient buffer management implemented in the PPSSM controls the packet drop and improves the throughput. If the number of receivers is 8 in the multicast group, then the proposed PPSSM protocol shows 11.6 % and 19.06 % improvement in throughput, over the existing mTreebone and CoolStreaming protocols respectively. Similarly, when the number of receiver nodes is increased to 1024 in the multicast group, due to the effective construction of the tree-mesh topology based on the degree of the node, network coding and buffer management, the proposed PPSSM protocol shows 30.11 % and 54.49 % improvement in throughput over the existing mTreebone and CoolStreaming protocols respectively.

5 Conclusion

This paper has proposed a novel PPSSM protocol, which uses an efficient tree-mesh construction, dynamic buffer management and network coding technique, to improve the throughput and packet loss. As the proposed protocol uses the network coding technique in addition to the effective tree-mesh construction, it shows improved performance over the other existing protocols, which did not use the network coding and dynamic buffer management techniques. When compared to the existing mTreebone protocol, the proposed PPSSM protocol shows 50.25 % improvement in reducing the packet loss rate, when varying the total number of overlay nodes in the tree-mesh topology from 16 to 2048 nodes. Also, the proposed protocol shows 30.11 % and 54.49 % improvement in throughput over the existing mTreebone and Coolstreaming protocols respectively, when the number of receiver nodes is 1024. Similarly, the proposed protocol shows improved performance in respect of the control overhead and average decoding time over the existing protocols.

References

1. Adams A, Nicholas J, SIadak W (2005) Protocol independent multicast - dense mode (PIM-DM): Protocol Specification (Revised), RFC 3973, http://www.ietf.org/rfc/rfc3973.txt

2. Banerjee S, Bhattacharjee B, Kommareddy C (2002) Scalable application layer multicast. Proc, ACM Sigcomm
3. Besharati R, Bag-Mohammadi M, Dezfouli MA (2010) A topology aware application layer multicast protocol. in Proc. CCNC,
4. Castro M, Druschel P, Kermarrec A, Nandi A, Rowstron A, Singh A (2003) Split-stream: high-bandwidth multicast in cooperative environments. in ACM SIGOPS. Oper Syst Rev 37:298–313
5. Cheng H, NaixueXiong, Vasilakos AV, Yang LT, Chen G, XiaofangZhuang (2012) Nodes organization for channel assignment with topology preservation in multi-radio wireless mesh networks. Ad Hoc Netw 10(5): 760–773 ISSN 1570-8705
6. Deering S, Cheriton D (1990) Multicast routing in datagram internetworks and extended LANS. ACM Trans Comp Syst 8(2):85–111[11] [RFC 3376] B. Cain
7. Deering S, Kouvelas I, Fenner W, Thyagarajan A (2002) Internet Group Management Protocol, Version 3. RFC 3376, http://www.ietf.org/rfc/rfc2236.txt
8. Demestichas PP, Stavroulaki VAG, Papadopoulou LM, Vasilakos AV, Theologou ME (2004) Service configuration and traffic distribution in composite radio environments. IEEE Trans Appl Rev 34(1):69–81
9. Fenner B, Handley M, Holbrook H, Kouvelas I (2006) Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601, http://tools.ietf.org/rfc/rfc4601.txt
10. Ho T, Lun DS (2008) Network Coding: An Introduction. In: Cambridge Univ. Press, Cambridge
11. Hosseini M, Ahmed DT, Shirmohammadi S, Georganas ND (2007) A survey of application-layer multicast protocols. Communications Surveys & Tutorials, IEEE, 9(3), 58–74, Third Quarter, doi:10.1109/COMST.2007.4317616.
12. Lee JJ, Tan T, Kakadia D, Delos Reyes EM, Lam MG. Dynamic setting of optimal buffer in IP networks, http://www.google.com/patents/US8223641.
13. Liang L, Song Y, Zhang H, Ma H, Vasilakos AV (2015) Physarum Optimization: A Biology-Inspired Algorithm for the Steiner Tree Problem in Networks. Comput, IEEE Trans on 64(3):819–832
14. Markou MM, Panayiotou C (2005) Dynamic control and optimization of buffer size in wireless networks. IEEE Veh Technol Conf 4:2162–2166
15. Padmanabhan V, Wang H, Chou P (2003) Resilient peer-to-peer streaming. proceedings of 11th IEEE international conference on network protocols, pp. 16–27. IEEE.
16. Peng Li, Song Guo, Shui Yu, Vasilakos, AV (2012) CodePipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. *INFOCOM*, 2012 Proc IEEE, 100–108.
17. Rudolf Ahlswede, NingCai, Shuo-Yen Robert Li, Raymond W Yeung (2000) Network Information Flow. IEEE Trans. on Inf. Theory, 46(4), 1204–1216,
18. Song Y, Liang L, Ma H, Vasilakos AV (2014) A Biology-Based Algorithm to Minimal Exposure Problem of Wireless Sensor Networks. IEEE Trans Netw Serv Manag 11(3):417–430
19. Thomos N, Frossard P (2010) Network coding of rateless video in streaming overlays. IEEE Trans Circ Syst Video Technol 20(12):1834–1847
20. Tran DA, Hua KA, Do T (2003) Zigzag: ancient peer-to-peer scheme for media streaming. In the IEEE computer and. Communications 2:1283–1292
21. Venkataraman V, Yoshida K, Francis P (2006) Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. Proceedings of the 14th IEEE International Conference on Network Protocols, 2–11,
22. Vidtorrent-vidtorrent (2013) URL http://web.media.mit.edu/~vyzo/vidtorrent/index.html. Accessed December 12, 2013.
23. Vlavianos A, Iliofotou M, Faloutsos M (2006) Bitos: Enhancing bittorrent for supporting streaming applications. In Proceedings – IEEE INFOCOM,
24. Wang F, Xiong Y, Liu J (2010) MTreebone: a collaborative tree-mesh overlay network for multicast video streaming. IEEE Trans Parallel Distri Syst 21(3):379–392
25. Youssef M, Ibrahim M, Abdelatif M, Chen L (2014) Vasilakos routing metrics of cognitive radio networks: a survey. IEEE Commun Surv Tutorials 16(1):92–109
26. Zhang X, Liu J, Li B, Yum TP (2005) DONet/CoolStreaming: a data-driven overlay network for peer-to-peer live media streaming. Proc IEEE INFOCOM 3:2102–2111
27. ZhijieShen, Luo J, Zimmermann R, Vasilakos AV (2011) Peer-to-Peer Media Streaming: Insights and New Developments. Proc IEEE 99(12):2089–2109
28. Zhou L, Chao H-C, Vasilakos AV (2011) Joint Forensics-Scheduling Strategy for Delay-Sensitive Multimedia Applications over Heterogeneous Networks. IEEE J Sel Areas Commun 29(7):1358–1367
29. Zhou L, Zhang Y, Song K, Jing W, Vasilakos AV (2011) Distributed Media Services in P2P-Based Vehicular Networks. Veh Technol, IEEE Trans on 60(2):692–703

T. Ruso received the B.Sc degree from Department of Computer Science, Aditanar College of Arts and Science, Tiruchendur, Tamilnadu, India, in 2002, the M.Sc degree from Department of Computer Science, Annamalai University, Tamilnadu, India, in 2005, the M.E degree from Department of Computer Science and Engineering, Crescent Engineering College, Chennai, India, in 2007. He did his Ph.D. in Anna University, Chennai, India in 2015. He is currently working as a senior grade Assistant Professor in Saveetha School of Engineering, Saveetha University, Chennai, India. His research interests includes Network Protocols, Application Layer Overlay Multicast, Multiparty Video Conference and Adhoc Network.

C. Chellappan received his Ph.D in the area of Database Systems from Anna University, Chennai, Tamil Nadu, India where he is currently the Principal of GKM College of Engineering and Technology and former senior professor in Department of Computer Science and Engineering, Anna University, Chennai, India. His research interest includes Mobile Computing, Sensor Networks and Parallel system scheduling. He is also a Principal Investigator of project on "Basic directed Collaborative Research in Smart and Secure Environment" sponsored by National Technical Research Organization, New Delhi, Govt of India, 2007-2011.

P. Sivasankar recive his Ph.D. in the area of wireless sensor networks. He is currently the Assistant Professor of Electrical and Engineering, NITTR, Chennai, India.