Robust Demand-Driven Video Multicast over Ad hoc Wireless Networks

D. Agrawal, T. Bheemarjuna Reddy, and C. Siva Ram Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras, India 600036
dvagr@cse.iitm.ernet.in, arjun@cs.iitm.ernet.in, murthy@iitm.ac.in

***Abstract*— In this paper, we address the problem of video multicasting in ad hoc wireless networks. The salient characteristics of video traffic make conventional multicasting protocols perform quite poorly, hence warranting application-centric approaches in order to increase robustness to packet losses and lower the overhead. By exploiting the path-diversity and the error resilience properties of Multiple Description Coding (MDC), we propose a Robust Demand-driven Video Multicast Routing (RDVMR) protocol. Our protocol uses a novel path based Steiner tree heuristic to reduce the number of forwarders in each tree, and constructs multiple trees in parallel with reduced number of common nodes among them. Moreover, unlike other on-demand multicast protocols, RDVMR specifically attempts to reduce the periodic (non on-demand) control traffic. We extensively evaluate RDVMR in the NS2 simulation framework and show that it outperforms existing single-tree and two-tree multicasting protocols.**

I. Introduction

An Ad hoc Wireless Network (AWN) is a collection of mobile nodes that dynamically form a temporary network without any pre-existing infrastructure. AWNs are characterized by high bit error rates and path breaks due to frequently changing network topology. This makes existing multicast protocols for wired networks unsuitable for AWNs, as a result of which several multicasting protocols for AWNs have been proposed in the literature. As developments in ad hoc networks continue, there is an increasing expectation with regard to content-rich multimedia (example video) communications in such networks, in addition to traditional data communications.

Video data is marked by

1) *High data rate* which leads to a considerable data overhead (which is defined as the total number of data packets transmitted/forwarded in the network).
2) *Soft real-time nature* A frame is considered lost if it has not received before its playback deadline at the receiver.
3) *Inter-frame dependency* In order to increase coding efficiency, motion compensated prediction (MCP) is used to exploit the temporal redundancy among successive video frames. This leads to a high degree of dependence among frames which effectively means that if some key frames are not received, an entire group of received frames can be rendered useless.

As shown in [1], [2], these characteristics make conventional multicast protocols proposed for AWNs perform quite poorly for video multicasting. This warrants development of application-centric video multicasting protocols which increase the error resiliency (i.e., robustness to packet losses) and reduce the total data overhead.

The recent advances in Multiple Description Coding (MDC) have made it highly suitable for multimedia applications in AWNs. MDC offers a way wherein we can split the video frame into multiple independent packets (also called descriptions), such that even if one of them is received the frame can be recovered although with degraded quality. This is indeed beneficial as the ability to even partially recover a frame can impact whether its subsequent received frames are decodable or not. We can further increase the error resilience by employing path-diversity (existence of multiple paths between a source-receiver pair) and sending each description along an independent path to the receiver. By making these paths as node-disjoint as possible, we decrease the correlation of path breaks among them, hence increasing the probability of atleast one description reaching the receiver.

In this paper, we exploit the error resilient properties of MDC along with path-diversity to propose a multiple tree protocol called Robust Demand-driven Video Multicast Routing (RDVMR) protocol. RDVMR reduces the data overhead by reducing the number of forwarders, using a novel path based Steiner tree heuristic. It constructs multiple trees in parallel, that have a reduced number of shared nodes (i.e., nodes that are forwarders for more than one tree). Therefore, each receiver has multiple independent paths to the source, along which different MDC descriptions are sent. To summarize, the three important contributions of this paper are:

1) We propose a novel path based Steiner tree heuristic to reduce the number of forwarding nodes and hence the total data overhead.
2) We propose RDVMR protocol which constructs multiple trees in parallel with a reduced number of shared nodes among them to provide robustness against path breaks.
3) Finally, we propose and evaluate a simple scheme wherein we use RDVMR to multicast different MDC descriptions over the different trees constructed.

The rest of the paper is organized as follows. In Section II we introduce the related work. In Section III we discuss the motivation behind our work. Section IV describes RDVMR in detail. In Section V we evaluate RDVMR through simulations, and finally in Section VI we conclude with possible future work.

II. Related Work

Multicasting in AWNs has been studied extensively. Various approaches to multicasting have been proposed in the literature. For an excellent taxonomy of them refer [3], [4]. Multicasting protocols disseminate data either by constructing a multicasting tree [5], [6], or by maintaining a mesh or a forwarding group [7], or through gossip based approaches [8]. The multicast routing protocols can either employ on-demand mechanisms like [5], [9], [10] or can rely on proactive (periodic) mechanisms like [11]. Recent works like [10] address the scalability issues of multicasting in AWNs.

Multicasting protocols may use Steiner trees [12] as the underlying distribution network to minimize the number of additional forwarders needed. Since constructing a Steiner tree is an NP complete problem, various polynomial time heuristics that approximate to a Steiner tree have been proposed and analyzed in wired networks [13]. However such centralized heuristics require the complete network topology to be maintained at the source. However these heuristics are not suitable for ad hoc wireless networks, because the routing overhead incurred to maintain the current topology at every node is quite high. Various distributed algorithms to construct Steiner trees have also been proposed [14], however they involve a high degree of message passing and hence may not converge in ad hoc network scenarios where the topology changes frequently due to the mobility of nodes. One of the popular path based greedy Steiner tree heuristics is the Shortest Path Tree (SPT) heuristic [15] which minimizes the number of extra nodes added to the tree each time a receiver joins. The authors of [15] proved that the competitive ratio of such a greedy approach is $O(logn)$, where n is the number of join-requests. Although various Steiner tree based multicast routing protocols for WANs [16] have been proposed, to the best of our knowledge, the problem of minimizing the number of forwarders in AWNs has not been studied before.

An automatic repeat request (ARQ) based loss recovery scheme for increasing the reliability would introduce unpredictable delays hence making it unsuitable for soft real-time traffic like video. Therefore to make video dissemination robust to packet losses, various data redundancy based techniques like Forward Error Correction (FEC) codes and Multiple Description Coding (MDC) have been proposed in the literature. Layered Coding (LC) based approaches are used to address the issue of receiver heterogeneity, wherein receivers have different video quality requirements based on their resource constraints. Recently several multipath unicast routing protocols have been proposed in the literature [17], [18]. Further, some attempts were made in the video unicasting by exploiting path diversity [19], [20], [21], [22], [23]. The authors of [19], [20], [22], [23] describe how MDC can use multiple paths to distribute descriptions over different paths for robust transmission. A comprehensive comparative study on MDC vs. LC for video streaming over multiple paths was done in [24]. The emphasis in all this work has been to use multiple paths for the video unicast. Robust multiple tree multicasting protocols for AWNs that exploit the global network topology information at the source to construct multiple trees are discussed in [6], [25]. The authors of [2] formulated video multicasting as an optimization problem in order to minimize the average perceived distortion by each receiver. Then a highly complex objective function using the global network topology was solved using genetic algorithm techniques. Although approaches like [2], [6], [25] give good results, they have a very high cost as they require an underlying link state routing protocol to provide them with the network topology. Unfortunately as link state routing [26] relies on flooding to disseminate the link update information, excessive control overhead may be generated, especially when high mobility triggers frequent link state updates. Hence these approaches are not suitable for realistic AWN scenarios.

Recently Wei and Zakhor proposed MDTMR [1], a protocol for video multicasting using MDC over multiple trees. It constructs two node-disjoint trees, one after the other. Each tree carries a MDC video stream. However it does not guarantee receiver connectivity to both trees, moreover it suffers from a tremendously high control overhead. On the other hand, our protocol constructs multiple trees in parallel with a low overhead. The multiple trees constructed have a reduced number of shared nodes and forwarders.

III. Motivation

In order to increase robustness, alternate approaches like gossip and mesh based multicasting could be considered. However gossip based approaches are unattractive as receivers may not receive frames sequentially, hence decoding of a video frame at the receiver could be delayed unless all previous frames it depends on indirectly are available. The high data rate of video traffic makes mesh based approaches unsuitable as the inherent redundancy increases the data overhead substantially. Moreover a mesh does not provide for any clean way to partition the video stream into multiple sub-streams (which is required if we are to send different descriptions over different multicasting structures). Hence in order to increase the robustness to packet losses, we employ path-diversity based approach. By path-diversity we mean that each receiver has different node-disjoint paths to the source, and hence we need to construct multiple trees (with reduced number of shared nodes) that connect the source to each of the receivers. Further we require that each of the multiple trees has reduced number of additional forwarders in order to decrease the overall data overhead.

Most of the existing multicasting protocols in AWNs construct shortest path trees, that do not optimize on the number of forwarders required. Having more forwarders leads to redundant data transmissions and unnecessarily depletes energy of uninterested nodes. This especially assumes significance for high data rate traffic like video, as the overhead of the redundant data transmissions becomes substantial. In fact most of the multicasting protocols do not even account for the data overhead, measuring just the control overhead. However as we show in Section V-B that the Normalized Packet Overhead (NPO) is a better metric to measure the total (data plus control) overhead of multicast routing protocols.

One possible approach to minimize the number of forwarders would be to collect the global topology at the source

by means of a link state routing protocol and employ a centralized Steiner tree heuristic, and then communicate the trees to each receiver [25]. In this approach, the receivers in a Steiner multicast tree have higher hop lengths to the source than they would in a shortest path tree. In video multicasting, the delay from the source to any particular receiver should be bounded. As discussed in Section II, the overhead incurred by link state routing is quite high, hence making centralized Steiner heuristics unsuitable for AWNs. The distributed algorithms given in [14] have a very high message passing overhead, take a long time to converge, and need to use beaconing for neighbor discovery. On the other hand, path based Steiner heuristics [27] are attractive as information about the path from the source to each node is readily maintained (for example by storing the entire path on a flooded packet originating from the source). In this paper we develop a novel path based Steiner tree heuristic to reduce the number of forwarders and hence the overall data overhead.

We use MDC because of its special property that all resulting descriptions are equally important, and reception of any one enables the decoding of that video frame. Therefore to increase the chances of even one description being received by a receiver, we must send them on different node-disjoint paths so as to make them robust against packet losses due to path breaks. We extend the path-diversity based approach for unicasting video [19] for multicasting by constructing multiple trees and then sending different descriptions over them.

The following points must be kept in mind while designing a multiple tree multicasting protocol for AWNs that reduces the number of forwarders and shared nodes.

- *Effect of dynamic joins/leaves and mobility* The multicast tree has to be updated and if necessary rearranged as nodes join and leave, or if they move to different locations. This rearrangement is necessary to maintain efficiency of the tree throughout the multicast session.
- *Minimizing number of forwarders* vs. *Increasing node disjointedness* It can be seen that as we increase the node disjointedness among multiple trees, we increase the number of forwarders. Hence there is a tradeoff between reducing the number of shared nodes among multiple trees and reducing the number of forwarders in each tree.
- *Shortest path* vs. *Steiner trees* Essentially Steiner based multicast trees have a higher number of downstream receivers served by each link, hence increasing the multicasting efficiency, which can be roughly defined as the number of receivers served by each link. However there is a flip side to increasing multicast efficiency, as in the presence of mobility, more number of receivers are susceptible to a single link failure. This could increase the number of packet losses and the repair control overhead with increasing mobility. Hence there is a tradeoff involved in reducing the number of forwarders versus increasing the PDR in presence of mobility.
- *Effect of increasing the number of descriptions and trees* We can clearly see that increasing the number of MDC descriptions and the number of trees would lead to better error resilience. Due to scarcity of network resources like bandwidth, energy, and computing power, the data rate cannot be arbitrarily increased, therefore the number of descriptions is bounded. The number of node-disjoint multicast trees that we can practically construct is limited by the node density, and hence constructing more than a few trees may not always be possible. Moreover increasing the number of trees leads to an increased control overhead, as multiple tree repair attempts would be happening simultaneously in the network.

IV. An overview of RDVMR

In order to make RDVMR as demand-driven as possible (i.e., to minimize the non on-demand control traffic), we base RDVMR on the Adaptive Demand Driven Multicast Routing (ADMR) protocol [9]. ADMR achieves very low NPO and high PDR compared to other conventional AWN multicasting protocols like ODMRP, ZMRP, and MAODV. ADMR is a receiver-initiated multicasting protocol, and it uses hard-state tree repair. Its salient features are: 1) There is no periodic control traffic like beaconing, or link state updates. 2) Most of its control information is piggybacked on data packets. 3) It is standalone, i.e., it does not require any underlying unicast routing protocol.

RDVMR builds multiple (say K) trees simultaneously, where each tree has lesser number of forwarders, and hence reduce the NPO, while maintaining the same PDR as achieved by the ADMR protocol.

A. Data Structures

The following data structures are maintained by each node:

- ***TreeState(t,groupId)*** This stores the parent $parent$ (next hop to the source) for the tree t of the multicasting group $groupId$. It also stores the last sequence number $lastSeq$ heard from the source through $parent$, and the cost $cost$ to reach the source through $parent$. Loosely speaking, each node knows of a next hop to the multicasting source, this next hop is known as the $parent$ of that node. All nodes, including nodes that are not part of any multicasting tree, keep track of their parents.
- ***PacketCache(s)*** This is a cache of recently heard packet sequence numbers originating from the source s. It is used to identify duplicate packets originating from the node s.
- ***TypeOfNode(t,groupId)*** This is a bit field which gives the type of a node for a particular tree t of the multicasting group $groupId$. It can be any one of FWD, RCV, NON_TREE, FWD_AND_RECV. FWD nodes are pure forwarders (i.e., they are not receivers), RCV nodes are receivers (but not forwarders), NON_TREE nodes are neither forwarders nor receivers, FWD_AND_RECV nodes are both forwarders and receivers.
- ***NodeState(d)*** This stores the unicast routing information for the destination d, namely the next hop $nxtHop$, the last sequence number heard $lastSeq$, and the cost $cost$ to reach node d through $nxtHop$. It is updated on hearing any packet originating from node d, similar to the backward routing protocol [28].

B. Protocol Overview

In this subsection we describe how RDVMR builds and maintains K trees. RDVMR is a tree-based receiver initiated multicasting protocol, where receivers join and leave on-demand. RDVMR adapts each tree to the continuously changing network topology and to the changing group membership. In order to maintain K trees, each packet contains the $treeId$, identifying the tree it is meant for, in addition to the $groupId$ (identifying the multicast group it is meant for). As an optimization, by maintaining the $treeId$ in each packet as a bit-vector of size K (named as $treeList$), a single control packet can be meant for multiple trees simultaneously. Note that, when we say that a node gets or sends a packet p along the tree t, we mean that $p.treeList$ has the t^{th} bit set. Hence by means of using a $treeList$ in each packet, RDVMR is able to build and maintain K trees in parallel.

The source of the group $groupId$ multicasts data packets along the tree. It also periodically floods a data packet having a piggybacked header called $SrcJoinAdvt$, which eventually reaches every node in the network. It advertises a route to the source of $groupId$ to all nodes in the network, and helps to refresh the multicast tree. Since the $SrcJoinAdvt$ packet advertises a route to the source along all trees, it has its $treeList$ set to all ones, whereas the packets multicasted by the source advertise a route to it only along a particular tree t. On hearing such a route advertisement to the source of the group $groupId$, a node creates (or refreshes) the entry $TreeState(t, groupId)$. This way every node keeps track of its upstream node (parent) for the tree t to reach the source. The source also keeps track of the mean inter-packet time ipt, which is piggybacked on every packet it sends. A tree node (i.e., a node that is a forwarder or a receiver) is said to be disconnected if it has not received any multicast packet within a small multiple of ipt. In order to keep the multicast tree intact, the source also multicasts keep alive packets every ipt if it does not have any data packets to send. Keep alive packets are multicasted along the tree just like data packets.

When a receiver wishes to join the tree t, it unicasts a $joinReq$ packet to its parent for the tree t. In case it does not have a parent, it floods a $MulticastSolicitation$ packet for the tree t. When a tree node (i.e., a node that is a forwarder or a receiver for tree t) gets a $MulticastSolicitation$, it unicasts it up the tree through its parent to the source. On receiving a $MulticastSolicitation$, the source unicasts a $joinReply$ packet to the corresponding receiver. A node forwarding a $joinReply$ packet to the node d for the tree t, becomes a forwarder for that tree.

If a forwarder for tree t detects a disconnection, it multicasts a $repairNotification$ packet downstream. A $repairNotification$ serves to inform the downstream nodes of the disconnection, and avoids redundant repair attempts by them when they eventually detect the disconnection. Simultaneously the forwarder detecting the break, attempts to locally repair the tree by flooding a limited ttl $LocalReconnect$ for the tree t. Similar to handling a $MulticastSolicitation$, a tree node unicasts a $LocalReconnect$ up the tree t to the source. The source then unicasts a $LocalReconnectReply$ to the node that originated the local repair. A $LocalReconnectReply$ is processed exactly like a $joinReply$, i.e., any node forwarding it becomes a forwarder for the tree t. In order to avoid routing loops we only allow the source to respond to flooded packets like $MulticastSolicitation$ and $LocalReconnect$. If a receiver detects a disconnection, it attempts to rejoin the group after some time by flooding a $MulticastSolicitation$ in case the repair attempts of its upstream nodes (i.e., nodes that are ancestors of this node along the multicast tree) fail. It can be clearly seen that the repair control overhead increases with mobility because of the increasing link breaks. The repair procedure presented in this paragraph is detailed in Algorithm 1.

RDVMR does not have explicit $leave$ messages, instead forwarders prune themselves away if they detect that they do not have any downstream children. Each multicasted data packet carries a field containing the parent of the node that forwarded it. This field serves to passively acknowledge the parent of that node to continue forwarding data. However, since pure receivers do not forward data packets, they need to explicitly acknowledge their parents by periodically unicasting an acknowledgment packet. This way a forwarder not having any downstream receivers prunes itself away from the multicasting tree.

Algorithm 1 $DisconnectionHandling$

```
node ← Node initiating the disconnection handling
if node is a forwarder in tree t then
    Multicasts a RepairNotification packet for tree t downstream
    Floods a limited ttl LocalReconnect packet for tree t
    Marks itself as disconnected and schedules expiration
end if
if node is a receiver in tree t then
    if node is NOT a forwarder in tree t then
        Floods a limited ttl LocalReconnect packet for tree t
    end if
    Schedules a network wide MulticastSolicitation flood
end if
```

C. Handling of routing packets

This subsection describes the handling of routing packets in RDVMR. RDVMR has three routing packets responsible for building and maintaining the multiple trees. They are $MulticastSolicitation$, $SrcJoinAdvt$, and $LocalReconnect$ packets originated by a receiver, source, or a forwarder for a particular group, respectively.

Each node in a multicast tree needs to learn of a loop free route to the source along that tree. In RDVMR similar to other multicast routing protocols, backward routing is employed. In backward routing, a node S floods a routing advertisement for itself (i.e., it sets $r.advertisedNode$ to S), and any node B hearing such an advertisement r from the node A, may choose A as the next hop to node S. A routing advertisement packet r contains a field $r.cost$ to store the cost of the path it has taken so far starting from its source (which is same as

$r.advertisedNode$) and a field $r.prevHop$ which is set to the node that forwarded r. This is illustrated in Fig. 1. In this example node S floods a routing advertisement packet r for itself, which is received by node B through node A (hence when node B receives r, $r.prevHop = A$).

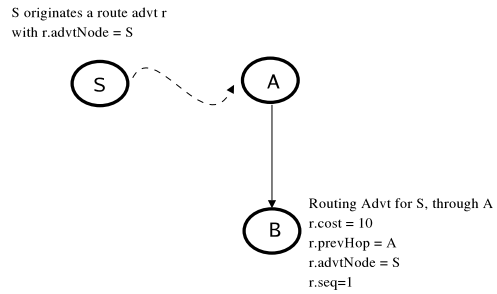

Fig. 1. Illustration of backward routing.

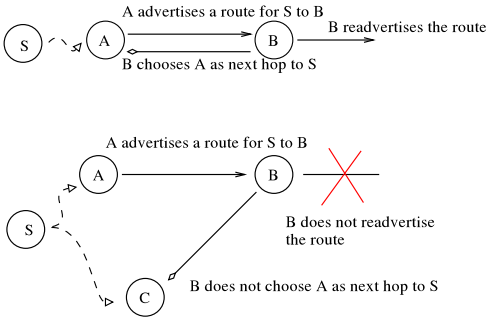

Fig. 2. Node B drops node A's routing advertisement for S as it does not choose it as the next hop to the node S.

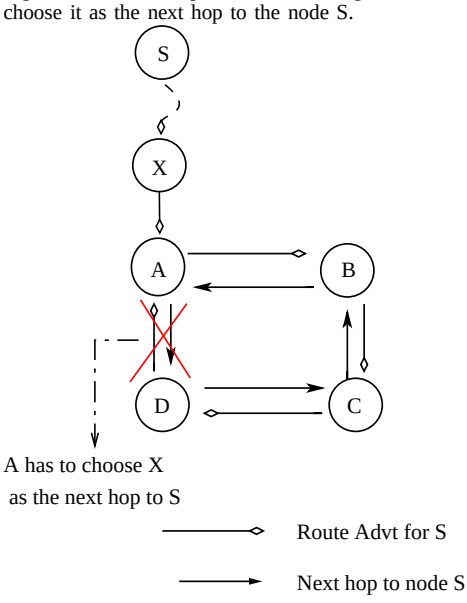

Fig. 3. Example scenario of a routing loop originating at node A.

We can ensure loop free paths if we impose the following two conditions:

- The cost in the routing advertisement packet monotonically increases. That is, in the above example the cost in the routing advertisement packet received by node B is more than the cost in the packet received by node A.
- A node is allowed to forward a routing advertisement r, only if it is received from it's next hop to the node $r.advertisedNode$. This is illustrated in Fig. 2, wherein node B only forwards the routing packet r if it chooses node A as its next hop to the node S.

We can easily prove using contradiction that the above two conditions guarantee loop free paths. Assume that even in the presence of these conditions routing loops can be formed. Such a scenario is illustrated in Fig. 3. Without loss of generality let us assume that node S originates the routing advertisement which is further forwarded by nodes X, A,B, C, and D. Let us assume that a routing loop is formed at node A. Since node B forwarded the routing advertisement it obtained through node A, node B must have chosen A as its next hop to S, similarly node C chooses node B as its next hop to S and node D chooses node C. As the cost carried in the routing advertisement monotonically increases, node A receives a higher cost route to node S from node D than it received from node X. Hence it should have chosen node X and not node D as its next hop to node S. Hence we have proved that it is impossible to form a routing loop if our routing protocol follows the above two conditions.

In order to adapt to the changing network topology, each routing packet r, must carry a monotonically increasing unique sequence number seq, to differentiate between stale and new routing information. Many conventional AWN routing protocols consider routing advertisement packets with a higher sequence number better than routing advertisements with a better cost. However, it can be clearly seen that such a routing protocol would converge to a shortest path routing protocol, as the newest routing advertisement packets for destination S are likely to first reach a node A along the shortest path from S to A.

In order to construct Steiner like trees (i.e., trees with reduced number of forwarders), the route from the multicast source S to any node A in the multicast tree should not be the shortest path from S to A. Therefore RDVMR not only needs to differentiate between stale and new routing advertisements, it must also make sure that it does not degenerate into finding shortest path trees. To achieve this, a node A on hearing a routing advertisement packet r for the node $r.advertisedNode$, may choose $r.prevHop$ as the next hop for $r.advertisedNode$, iff $(r.cost < NodeState(r.advertisedNode).cost)$ AND $(r.seq >= NodeState(r.advertisedNode).lastSeq)$. Essentially a node chooses a neighbor to be a next hop for a destination, only if it has received a routing advertisement for that destination from that neighbor which has a higher (hence newer) sequence number and a better cost. The cost field $r.cost$ is updated by the cost function (refer Algorithm 2).

Note that a node must forward a routing advertisement from a neighbor, if it chooses that neighbor as the next hop to its destination. Doing this ensures that every node in the network learns of the best route to a particular destination. Therefore a node must forward its best learned routing advertisement packets even if those routing packets have been seen by the node before. That is if a node chooses $r.prevHop$ as the next hop to $r.advertisedNode$ on hearing the routing advertisement

packet r, it must forward r again, even if it has already seen (and hence forwarded) $r.seq$ before when it received it through some other node X (not equal to $r.prevHop$). This obviously leads to a high routing control overhead as duplicate routing packets are also forwarded in case they have a better cost. Typical AWNs have network diameters of less than around ten hops, hence by setting a threshold on the number of times a duplicate routing packet may be forwarded if it has a better cost (but is duplicate) achieves a good tradeoff between finding good routes to the destination versus the control overhead. Any node that forwards a duplicate but better routing advertisement packet r increments the field $r.dupButBetterCount$ in r. When the value of this field reaches a certain threshold, r is dropped.

As described above, a routing packet r in RDVMR can either be a $MulticastSolicitation$, or a $SrcJoinAdvt$, or a $LocalReconnect$ packet. In order to maintain upto date routing state, the rate of sending these routing packets must keep in step with the mobility. RDVMR uses a mixture of proactive and reactive routing; the source periodically floods $SrcJoinAdvt$ to proactively refresh the routing state, whereas the $MulticastSolicitation$ and $LocalReconnect$ packets update the routing state in a reactive manner (as link breaks happen).

D. Steiner tree heuristic

In this subsection we describe our novel path-based Steiner tree heuristic, which tries to reduce the number of additional forwarding nodes required by differentially costing each node along each tree. The details of the proposed heuristic are abstracted into a cost function $CostOfNode$, by means of which we are able to impose a routing gradient so as to heuristically reduce the number of forwarders. Note that we differ from shortest path routing, in which the cost function is simply the hop count to the source. By differentially costing receivers, forwarders, and non tree nodes, and by encouraging links to serve more downstream receivers, we can reduce the number of forwarders, hence leading to a decreased NPO.

The basic motivations behind the proposed cost function are:

- **Increasing the number of downstream receivers under each link** By assigning a lesser cost to a forwarder with more number of downstream children under it, we encourage an increase in the multicast efficiency (the bushyness of the tree).
- **Making as many receivers as forwarders** By assigning a lesser cost to receivers, we can lower the number of extra forwarding nodes needed per tree.
- **Decreasing the number of forwarders** Similar to other SPT Steiner heuristics [14], we assign a higher cost to non tree nodes, hence promoting paths to the source which have the lowest branch cost to the tree.
- **Increasing the node disjointedness of the K trees** In order to reduce the number of shared nodes among different trees, the cost also includes a disjointedness component, which is higher for nodes being forwarders for multiple trees, and zero for nodes which are a part of just one tree. Note that this conflicts with reducing the number of forwarders, and a tradeoff between them is achieved by the parameters γ and λ in the cost function.

Each node adds its cost as returned by the function $CostOfNode$ to the cost fields in the routing packets (namely $SrcJoinAdvt$, $MulticastSolicitation$, and $LocalReconnect$). The following psuedocode describes the function $CostOfNode$.

Algorithm 2 $CostOfNode$

$node \leftarrow$ Node that is calculating its cost
$groupId \leftarrow$ Multicast group address
$baseCost \leftarrow$ NON_PART_OF_TREE_COST
$treeId \leftarrow$ Tree along which cost is being calculated
if $node$ is a receiver for tree $treeId$ of multicast group $groupId$ **then**
  $baseCost = baseCost * \alpha$
else if $node$ is a forwarder for tree $treeId$ of multicast group $groupId$ **then**
  $ncr \leftarrow numReceiverDescendants(treeId, groupId)$
  $baseCost \leftarrow (baseCost * \beta)/ncr$
end if
if $numTreesPartOf(groupId) > 1$ **then**
  $disjointedCost \leftarrow$
  $\lambda * NON\_PART\_OF\_TREE\_COST$
  $*(numTreesPartOf(groupId) - 1)$
  return $\gamma * (disjointedCost) + (1 - \gamma) * baseCost$
else
  return $baseCost$
end if

The function $numTreesPartOf(groupId)$ returns the number of trees this node is a forwarder of for the group $groupId$. The function $numReceiver-Descendants(treeId, groupId)$ returns the number of receiver descendants in the subtree rooted at $node$ for the tree $treeId$ of the group $groupId$. We keep track of the number of receiver descendants under a node for a particular tree by piggybacking this information on join requests and passive acknowledgments for that particular tree.

We set α to be much lower than β, because we want a receiver to have a much lower cost than a forwarder. Both α and β are less than one. The parameters γ and λ control the tradeoff between the node disjointedness and the number of forwarders, as discussed earlier.

We try to reduce the number of forwarders by letting this cost function guide the routing in RDVMR. By greedily making every path from the source of the multicasting tree to a particular node in the tree have less number of forwarders, we are able to reduce the total number of forwarders in the tree. Although, note that this greedy approach that utilizes only the path information is not able to reduce the number of forwarders much compared to the more expensive Steiner tree algorithms mentioned in [14], which work with the global network topology, and are hence effective in identifying candidate Steiner nodes. However unlike SPT heuristic, we also take into account the properties of the path from the

source to the last tree node (to which SPT would connect a new node to), and hence do not simply minimize the number of forwarders added at each step. As shown in Section V-C.3, that this less greedy approach performs better than SPT.

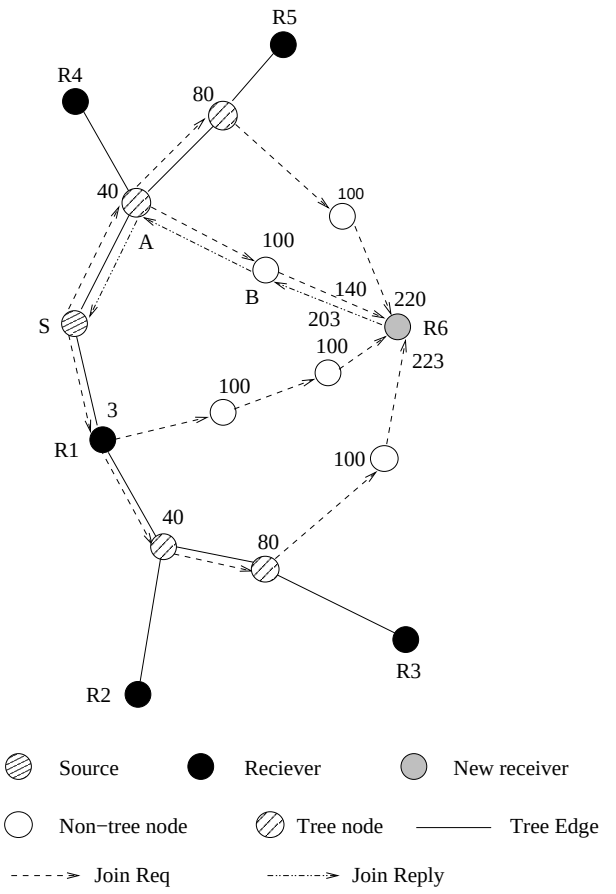

Fig. 4. An example to illustrate the cost function.

The cost function is illustrated in Figure 4. In this example, source S floods a join request having a piggybacked cost. Node $R6$ is the new receiver. The join requests arrive at node $R6$ through four paths, of which the one through A and B is chosen as it has the least cost. In this example $NON\_PART\_OF\_TREE\_COST = 100$, $\alpha = 0.03$, $\beta = 0.8$, and $\gamma = 0$. The cost of each node and of the join requests received by node C along different paths are shown in the figure.

V. Simulation Studies

A. Simulation framework

We implemented RDVMR in the NS2 simulation framework [29]. We compare its performance with ADMR and MDTMR [1]. ADMR has been shown to perform better than other conventional multicasting protocols like ODMRP, and hence it was chosen for comparison. MDTMR is a two-tree video multicasting protocol.

We first compare our protocol for the single tree case with ADMR for various scenarios, and then using multiple trees we compare our performance with MDTMR. These performance results are collected by running multiple simulation runs. Each run of the simulator executes a scenario containing all of the movement behavior of the ad hoc network nodes and all application-layer communication originated by the nodes, generated in advance so that it can be replayed identically for the different routing protocols and variants studied. Each routing protocol studied was thus challenged by an identical workload. For all experiments we set the parameters α, β, γ, and λ in RDVMR to be 0.03, 0.8, 0.2, and 2, respectively. These values were observed to give good results for typical AWN settings. We will briefly explain the functioning of MDTMR in the subsequent paragraph.

MDTMR constructs two node-disjointed trees in a serial manner. Using MDC it codes each video frame into two descriptions. Each description is sent along one tree. A frame is marked undecodable if none of the two descriptions can be received before its playback deadline. MDTMR measures the percentage of undecodable frames over the total number of video frames sent out by the source. The trees are numbered as $T1$ and $T2$. In MDTMR the source periodically floods a $joinReq$ packet for tree $T1$, receivers reply with $joinAcks$ along the reverse path to the source. The $joinAcks$ are relayed to the source by the intermediate nodes. On relaying a $joinAck$ for a tree, a node becomes a forwarder for that tree, and stops forwarding $joinReq$ for the other tree. Hence if a node becomes a forwarder for tree $T1$, it does not forward the flooded $joinReq$ for tree $T2$. The source floods a $joinReq$ for tree $T2$ on receiving a $joinReply$ for tree $T1$ (and vice versa). This way MDTMR constructs disjointed trees. There is no explicit repair, tree states are refreshed by periodic flooding of $joinReq$ packets. This leads to a very high overhead. Also MDTMR may fail to ensure connectivity to all receivers, there may be some receivers that are not connected to the source by both trees.

B. Metrics

We evaluate the performance using the following metrics:

1) **Percentage of forwarders**: This metric is defined as the percentage of number of pure forwarders to the total number of nodes in the tree. This measures the efficiency in terms of minimizing the number of forwarding nodes.
2) **Packet Delivery Ratio**: It is defined as the average of the ratio of the number of data packets received by each receiver over the number of data packets sent by the source.
3) **Normalized Packet Overhead**: It is defined as the ratio of the total number of packets (control and data) exchanged in the network over the total number of data packets received by all the receivers. This measures both the data forwarding efficiency and also the control overhead of the multicasting protocol.
4) **Ratio of undecodable frames**: It is defined as the average of the ratio of number of undecodable frames experienced by each receiver over the total number of frames it should have received. A frame is said to be undecodable if none of its descriptions have been received before its playback deadline or if the frame it depends on is undecodable. This metric is more

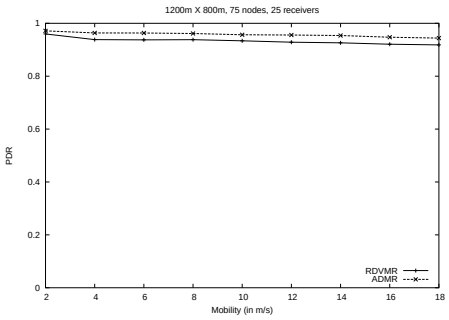

Fig. 5. Variation of PDR vs. Mobility.

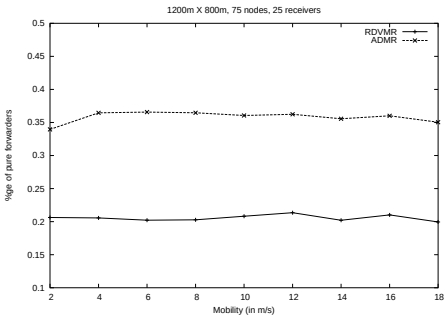

Fig. 6. Variation of %ge of pure forwarders vs. Mobility.

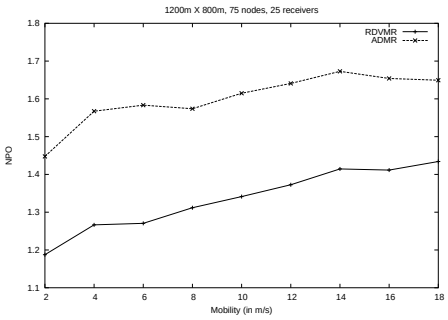

Fig. 7. Variation of NPO vs. Mobility.

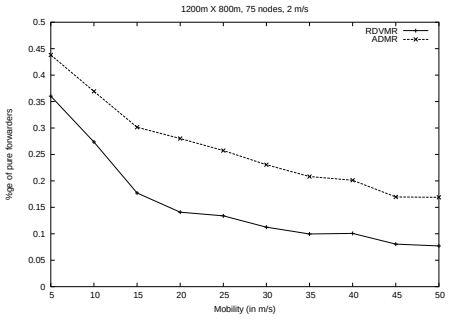

Fig. 8. Variation of %ge of pure forwarders vs. Number of receivers.

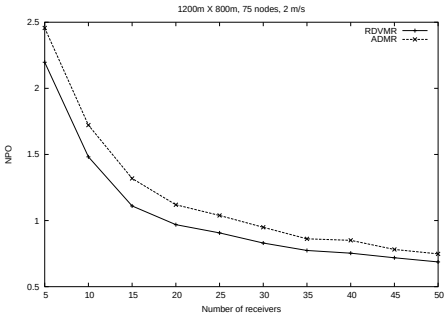

Fig. 9. Variation of NPO vs. Number of receivers.

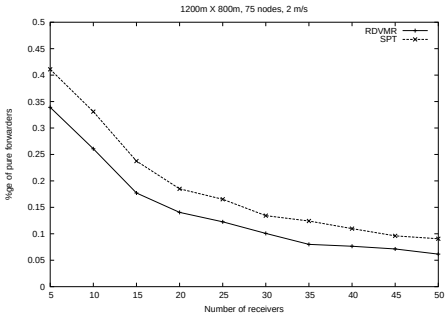

Fig. 10. Variation of %ge of pure forwarders vs. Number of receivers.

indicative of the quality of video received as it takes into account the delay sensitive nature of video frames, the inter-frame dependencies and it also highlights the fact that MDC helps to conceal frame losses caused by missing packets.

5) **Control overhead**: It is defined as the ratio of the total number of control packets exchanged in the network over the total number of successfully decoded video frames at each receiver.

C. Simulation results

1) Simulation scenario: We implemented RDVMR in the NS2 version 2.1b8. The IEEE 802.11 DCF is used as the underlying MAC layer protocol. The radio model is based on the Lucent/Agere WaveLAN/OriNOCO IEEE 802.11 product, which is a shared-media radio with a transmission rate of 2 Mbps, and a radio range of 250 meters. Our simulation setup consists of 75 nodes randomly spread in a rectangular terrain of area 1200X800 m^2. Each simulation runs for a period of 900 seconds. There is just one session and the source sends data throughout the simulation period. All the results presented in this paper were averaged over 30 simulation runs. 25 of the total nodes are randomly chosen to be receivers. Each of these receivers joins at a random time instant, chosen uniformly from $(4, 450)$ seconds. The receivers do not leave the multicast session. The source is a CBR source sending 512 byte sized packets at a rate of four packets per second unless otherwise stated. We used the Random Way Point model for simulating the mobility of nodes. Each node moves with some constant speed (i.e., min speed is equal to max speed) with zero pause time. The playback deadline is 150 ms, if a packet is not received within its playback deadline it is considered lost. Unless otherwise stated we use this simulation setup for our experiments.

2) Comparison with ADMR for a single tree: We first compare the performance of RDVMR with that of ADMR for a single tree. For this experiment, we used a simulation setup as described in Section V-C.1. We study the behavior of RDVMR and ADMR as the mobility is increased from 2 m/s to 18 m/s. Similar to ADMR, we fix the periodicity of flooding $SrcJoinAdvt$ to be 30 seconds. Since the multicast tree changes with time, the structural properties are time averaged over the entire simulation run. The structural metrics were got by taking periodic snapshots of the time varying multicast tree at every second and then averaging the values obtained. Figures 5 to 7 illustrate our results. It can be seen from Fig. 5 that the PDR of both protocols decreases with mobility, however RDVMR achieves almost the same PDR as that of ADMR at a lesser cost (refer Fig. 6) in terms of the number of forwarders. It can be seen from Fig. 6 that the number of forwarders is independent of the mobility. This reduction in number of forwarders reduces the NPO as can be seen in Fig. 7.

3) Evaluating the effectiveness of the cost function: In this experiment we study the effectiveness of our cost function as the number of receivers is increased. We used the simulation setup described in Section V-C.1 with a node mobility of 2 m/s. The number of receivers was varied from 5 to 50. When the randomly chosen receivers are quite far apart (as is the case when the number of receivers is less compared to the number of nodes in the network), there is less scope for reducing the number of forwarders. As can be seen in Fig. 8 both ADMR

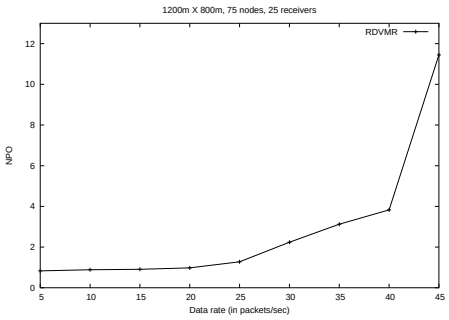

Fig. 11. Variation of NPO vs. Data rate.

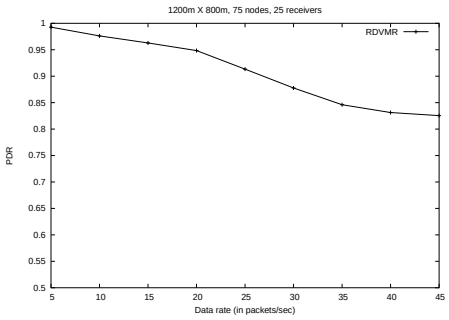

Fig. 12. Variation of PDR vs. Data rate.

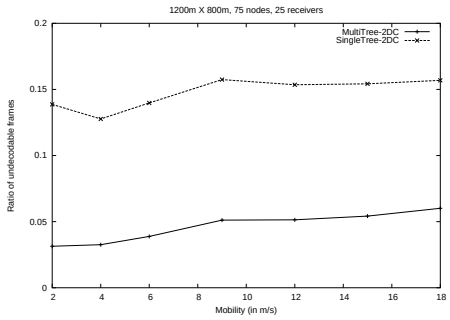

Fig. 13. Variation of Ratio of undecodable frames vs. Mobility.

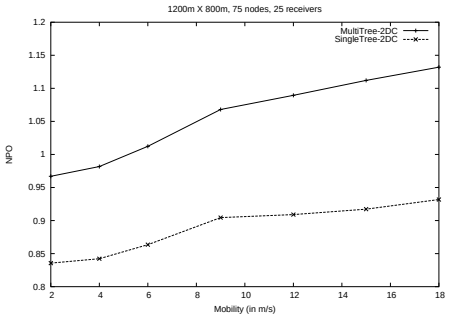

Fig. 14. Variation of NPO vs. Mobility.

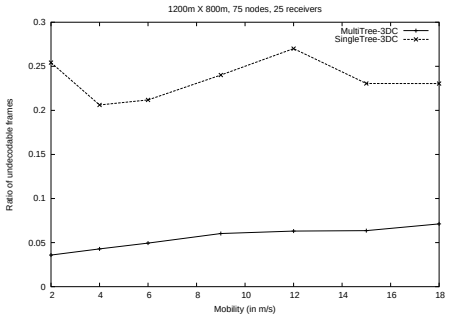

Fig. 15. Variation of Ratio of undecodable frames vs. Mobility.

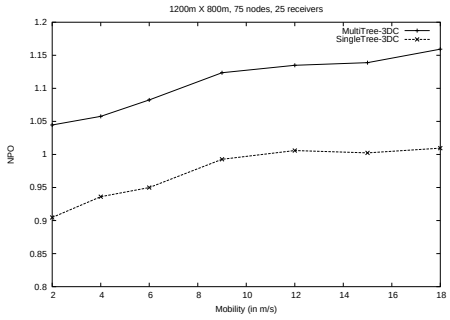

Fig. 16. Variation of NPO vs. Mobility.

and RDVMR cannot reduce the number of forwarders much in such scenarios. The proposed cost function is more effective as the number of receivers increases. The reduced number of forwarders dictates a falling NPO as can be seen in Fig. 9. As explained in Section IV-D, we are unable to reduce the number of forwarders much as we work with lesser topology information than other non path based Steiner tree heuristics.

We next compare RDVMR's cost function with SPT. We used the same simulation setup as used in the previous experiment. Fig. 10 shows that RDVMR's cost function performs marginally better than SPT in terms of reducing the number of forwarders. This is because SPT does not take into account the already constructed subtree, on the other hand RDVMR encourages nodes to connect to those subtrees that have a higher bushiness and more number of receivers. It can be seen that this less greedy approach performs better than SPT in the long run.

4) Evaluating RDVMR's scalability with respect to increasing data rates: Next we evaluate RDVMR's scalability with respect to increasing data rates. For this experiment the simulation setup of Section V-C.1 was used with the source varying its sending rate from 5 to 45 packets per second and with a static scenario. It can be seen in Fig. 11 and Fig. 12 that RDVMR scales decently till about 32 packets per second but beyond that it is unable to handle the increased load. The reason for this is that with increasing data rates more data can be lost for the same amount of time for which the path is broken. Hence the PDR drops with increasing data rate. This experiment is important because the data rate is directly proportional to the number of MDC descriptions used. By increasing the number of descriptions we can reduce the ratio of undecodable frames, however the results of this experiment dictate that for a video frame rate of 8 fps, we should not consider more than 4 descriptions (i.e., 32 packets/sec).

5) Merits and demerits of using multiple trees for MDC video multicasting: In our subsequent experiments, we study the advantages and disadvantages of using multiple trees for MDC video multicasting. We used the same simulation setup of Section V-C.1 while varying the node mobility from 2 m/s to 18 m/s. We conduct two experiments to evaluate the effectiveness of using two and three trees, each carrying a single description compared to sending multiple descriptions on a single tree. Each description has a rate of 8 packets/second.

In our first experiment we compare two schemes carrying two MDC descriptions: *SingleTree-2DC* and *MultiTree-2DC*. The SingleTree-2DC scheme carries both MDC descriptions on a single tree with a total rate of 16 packets/second. The MultiTree-2DC scheme uses two trees, each description is sent over a different tree with a per tree rate of 8 packets/second. Thus in both cases, the total rate is equal to 16 packets/second. It can be seen from Fig. 13 that the ratio of undecodable frames is reduced in scheme MultiTree-2DC compared to scheme SingleTree-2DC. This is because by sending each description along a separate tree we decrease the loss correlation among the two descriptions. It can be seen from Fig. 14 that the NPO for scheme MultiTree-2DC is less than double the NPO of the scheme SingleTree-2DC. Hence it can be seen that sending each of the two descriptions on different independent trees is advantageous.

In the second experiment we consider two schemes carrying three MDC descriptions: *SingleTree-3DC* and *MultiTree-3DC*. The SingleTree-3DC scheme carries all the three MDC

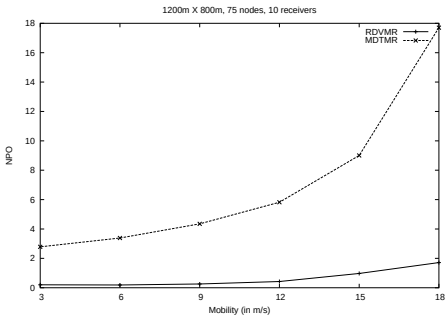

Fig. 17. Variation of NPO vs. Mobility.

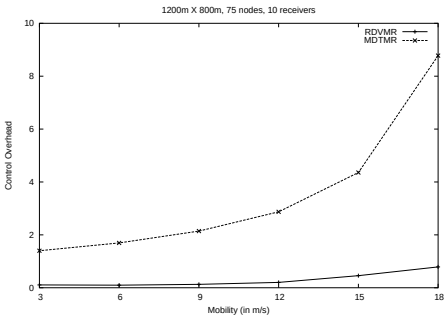

Fig. 18. Variation of Control Overhead vs. Mobility.

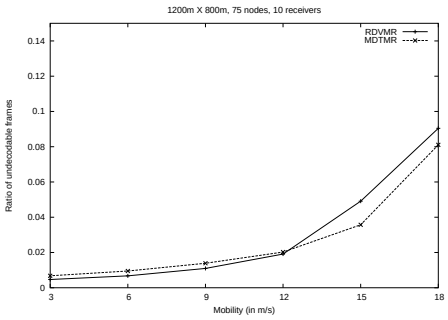

Fig. 19. Variation of Ratio of undecodable frames vs. Mobility.

descriptions on a single tree with a total rate of 24 packets/second. The MultiTree-3DC scheme uses three trees, each description is sent over a different tree with a per tree rate of 8 packets/second. Thus in both cases, the total rate is equal to 24 packets/second. Similar to the previous experiment, it can be seen from Fig. 15, that the ratio of undecodable frames is reduced in scheme MultiTree-3DC compared to scheme SingleTree-3DC. It can be seen from Fig. 16 that the NPO for scheme MultiTree-3DC is less than triple the NPO of the scheme SingleTree-3DC.

6) Comparison of RDVMR using two trees with MDTMR: We next compare RDVMR's performance to MDTMR [1]. We use the simulation scenario as mentioned in Section V-C.1. The node mobility varies from 3 m/s to 18 m/s and the number of receivers is set to 10. The $joinReq$ flooding interval in MDTMR was fixed to 3 seconds (as mentioned in [1]). Both MDTMR and RDVMR send two MDC descriptions on two trees (i.e., one description per tree). Each description has a data rate of 8 packets/second. As mentioned earlier MDTMR uses a naive flooding based tree construction, which has a very high overhead as can be seen in Fig. 17 and Fig. 18. We observe from Fig. 19 that RDVMR achieves almost the same ratio of undecodable frames as MDTMR at a tremendously lower cost. Moreover unlike MDTMR, RDVMR does not construct entirely node disjointed trees therefore every receiver can be connected to the source by both the trees.

VI. Conclusion and Future work

In this paper we proposed an effective low overhead, multi-tree based video multicasting protocol that exploits path-diversity along with the error resilience properties of MDC to achieve an improved performance over single and multi-tree protocols. Simulation results showed that RDVMR outperformed ADMR and MDTMR in terms of NPO and number of additional forwarding nodes required. We also observed that being less greedy helps in reducing the number of forwarding nodes in the long run compared to other Steiner heuristics like SPT.

For our future work, we seek to explore novel ways of allocating the MDC descriptions over multiple trees and also on how to assign them different rates for better perceived video quality at the receiver. We also wish to address the congestion and receiver heterogeneity issues of video multicasting.

References

[1] W. Wei and A. Zakhor, "Multipath Unicast and Multicast Video Communication over Wireless Ad hoc Networks", *in Proc. of BROADNETS 2004*, pp. 496-505, October 2004.

[2] Shiwen Mao, Xiaolin Cheng, Y. Thomas Hou, and Hanif D. Sherali, "Multiple Description Video Multicast in Wireless Ad hoc Networks", *in Proc. of BROADNETS 2004*, pp. 671-680, October 2004.

[3] C. Siva Ram Murthy and B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall, New Jersey, NY, USA, 2004.

[4] C. E. Perkins, *Ad Hoc Networking*, Addison Wesley Proffesional, New York, NY, USA, 2000.

[5] E. M. Royer and C. E. Perkins, "Multicast operation of the Ad-hoc On-Demand Distance Vector Routing Protocol", *in Proc. of ACM MOBICOM 1999*, pp. 207-218, August 1999.

[6] Sajama and Zygmunt J. Haas, "Independent-tree Ad hoc Multicast Routing (ITAMAR)", *Mobile Networks and Applications*, vol. 8, no. 5, pp. 551-566, October 2003.

[7] Sung Ju Lee, William Su, and Mario Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks", *Mobile Networks and Applications*, vol. 7, no. 6, pp. 441-453, December 2002.

[8] J. Luo, P.T. Eugster, and J.-P. Hubaux, "Route driven Gossip: Probabilistic Reliable Multicast in Ad hoc Networks", *in Proc, of IEEE INFOCOM 2003*, vol. 3, pp. 2229-2239, April 2003.

[9] J. G. Jetcheva and D. B. Johnson, "Adaptive Demand-driven Multicast Routing in Multi-hop Wireless Ad hoc Networks", *in Proc. of ACM MobiHoc 2001*, pp.33-44, October 2001.

[10] Lusheng Ji and M. S. Corson, "Explicit Multicasting for Mobile Ad hoc Networks", *Mobile Networks and Applications*, vol. 8, no. 5, pp. 535-549, October 2003.

[11] P. Sinha, R. Sivakumar, and V. Bharghavan, "MCEDAR: Multicast Core-extraction Distributed Ad hoc Routing", *in Proc. of IEEE WCNC 1999*, pp. 1313-1317, September 1999.

[12] P. Winter, "Steiner Problem in Networks: a Survey", *Networks*, vol. 17, no. 2, pp. 129-167, Summer 1987.

[13] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees", *Acta Informatica*, vol. 15, pp. 141-145, 1981.

[14] F. Bauer and A. Varma, "Distributed Algorithms for Multicast Path Setup in Data Networks", *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 181-191, April 1996.

[15] M. Imase and B. M. Waxman, "Dynamic Steiner Tree Problem", *SIAM Journal on Discrete Mathematics*, vol. 4, no. 3, pp. 369-384, August 1991.

[16] Sriram Raghavan, G. Manimaran, and C. Siva Ram Murthy, "A Rearrangeable Algorithm for the Construction of Delay-constrained Dynamic Multicast trees", *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 514-529, August 1999.

[17] S.-J. Lee and M. Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks", *in Proc. of IEEE ICC 2001*, vol. 10, pp. 3201-3205, June 2001.

[18] A. Nasipuri and S. R. Das, "On-demand Multipath Routing for Mobile Ad hoc Networks", *in Proc. of IEEE ICCCN 1999*, pp. 64-70, October 1999.

[19] Shiwen Mao, Shunan Lin, Yao Wang, S. S. Panwar, and Yihan Li , "Multipath Video Transport over Ad hoc networks", *IEEE Wireless Communications*, vol. 12, no. 4, pp. 42-49, August 2005.

[20] Shiwen Mao, Shunan Lin, S. S. Panwar, Yao Wang, and E. Celebi, "Video Transport over Ad hoc Networks: Multistream Coding with Multipath Transport", *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1721-1737, December 2003.

[21] El Al, A. A. Saadawi, and T. Myung Lee, "Improving Interactive Video in Ad-hoc Networks using Path Diversity", *in Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems 2004*, pp. 369-378, October 2004.

[22] J. G. Apostolopoulos, "Reliable Video Communication over Lossy Packet Networks using Multiple State Encoding and Path Diversity", *in Proc. of Visual Communication and Image Processing 2001*, pp. 392-409, January 2001.

[23] S. Somsundaram, K. P. Subbalakshmi, and R. N. Uma, "MDC and Path Diversity in Video Streaming", *in Proc. of IEEE ICIP 2004*, vol. 5, pp. 3153-3156, October 2004.

[24] J. Chakareski, S. Han, and B. Girod, "Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths", *in Proc. of ACM Multimedia*, pp. 422-431, November 2003.

[25] G. H. Lynn and T. F. Znati, "Robust Multicasting using an Underlying Link State Unicast Protocol", *in Proc. of HICSS 2004*, p. 90293b, January 2004.

[26] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol for Ad hoc Networks", *in Proc. of IEEE INMIC 2001*, pp. 62-68, December 2001.

[27] Pawel Winter and J. MacGregor Smith, "Path-distance Heuristics for the Steiner Problem in Undirected Networks", *Algorithmica*, vol. 7, no. 2-3, pp. 309-327, 1992.

[28] Yogen K. Dalal and Robert M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets", *Communications of the ACM*, vol. 21, no. 12, pp. 1040-1048, December 1978.

[29] ns-2: network simulator, *http://www.isi.edu/nsnam/ns/*