Abstract—This paper addresses the problem of video multicast for heterogeneous destinations in mobile ad hoc networks. Multiple Description Coding (MDC) is used for video coding. MDC generates multiple independent bit-streams, where the multiple bit-streams are referred to as multiple descriptions (MD). Furthermore, MDC enables a useful reproduction of the video when any description is correctly received. Specifically, we propose Sequential MDMTR (Multiple Disjoint Multicast Trees Routing) protocol for video multicast. It builds multiple disjoint multicast trees and assigns MD video in a centralized fashion. Sequential MDMTR protocol aims at increasing the number of assigned MD video to each destination. We extensively evaluate our proposed protocol by simulations and show that it outperforms the existing work.

Keywords—MANETs; Video multicast; MDC;

I. INTRODUCTION

Mobile ad hoc network (MANET) is a self-organized and dynamically reconfigurable wireless network without central administration and wired infrastructure. Nodes in MANET can instantly establish a communication structure while each node moves in an arbitrary manner. Thus MANET is useful for mobile nodes working in a group to accomplish a certain task.

Video multicast over wireless ad hoc networks has been studied in recent years [1], [2], [3], [4]. The main objective of these studies is to improve the quality of the received video by exploiting the error resilience properties of MDC along with multiple paths. In other words, MD video are encoded and transmitted over different paths to each destination. If any path is broken, packets corresponding to the other descriptions on the other paths can still arrive at the destination node on time. However, these protocols are developed under the assumption that destination nodes wish to receive all the video descriptions sent by the multicast source, i.e., they do not support heterogeneous destinations. However, authors in [5], [6] proposed different algorithms and protocol for video multicast over wireless ad hoc networks. They deployed the independent-property of MDC to improve the user’s satisfaction for a group of destinations.

In this paper, we address the problem of video multicast for heterogeneous destinations and we propose Sequential-MDMTR protocol. In Sequential-MDMTR protocol the construction of multiple disjoint multicast trees and the assignment of MD video are performed in a centralized manner, i.e., by the multicast source. Sequential-MDMTR requires that all destination nodes must be on the first multicast tree to receive the first video description (MDC1), and then the destination nodes that require the second video description (MDC2) must be on the second tree, and so on, hence the name sequential. For this purpose, we propose an efficient algorithm for constructing multiple multicast trees and assigning MD video. Then based on this algorithm we develop our proposed protocol.

The remainder of the paper is organized as follows. Section II presents our problem formulation. We present our proposed protocol Sequential MDMTR in Section III. In Section IV, we evaluate our proposed protocol through simulations, and finally in Section V we conclude the paper.

II. PROBLEM FORMULATION

Our video multicast problem can be formulated as follows. Given a network \( G = (V, E) \) where \( V \) is the set of vertices representing wireless nodes, and \( E \) is the set of edges representing wireless links. A wireless link between two nodes indicates that both nodes are within the transmission range of each other. A multicast source \( S \) in the network transmits video to \( m \) destination nodes given by the set \( \mathbb{R} = \{ R_1, R_2, \ldots, R_m \}, \mathbb{R} \subseteq V - S \), with corresponding bandwidth requirements \( \mathcal{B} = \{ b_1, b_2, \ldots, b_m \} \). Then, find a multicast tree(s) rooted at the multicast source \( S \) and spanning the destination set \( \mathbb{R} \) such that the total number of assigned video descriptions to each destination \( R_i \), defined in (1), is maximized.

\[
\text{maximize} \{ N_{\text{asg}}(R_i) \} \quad (1)
\]

where \( N_{\text{asg}}(R_i) \) is the number of the assigned video descriptions to the destination \( R_i \).

III. SEQUENTIAL MDMTR PROTOCOL

Sequential MDMTR is an on-demand video multicast routing protocol that constructs multiple multicast trees and assigns MD video to the multicast trees in a centralized way. The construction of multiple multicast trees and the assignment of MD video are performed using three-way handshaking approach (Route Request (RouteReq), Route
A. Data structure

- Multicast routing table: Each node creates and maintains a Routing Table for the tree $t$. It stores the source address, multicast group groupID, the addresses of the upstream and downstream nodes, minimum hop count from the source, and last sequence number heard from the source through the upstream node.

- Membership table: The Multicast group information is stored in the Membership Table that is created and maintained by each node for the tree $t$. The Membership Table contains the multicast group address, nodes status (multicast source, destination, pure forwarder).

- Message cache: The message cache is generated and maintained by each node to detect duplicated packets.

- Timeslot table: Each node creates and maintains a table that contains the status of timeslots (free, reserved, candidate).

B. Route Discovery

When a multicast source receives a request from the application layer to set up a multicast connection to a group of destination nodes with bandwidth requirements for each video description, it initiates a RouteReq packet and floods it to its neighbors, as seen in Fig. 1(a). The RouteReq packet contains the following fields: (source, request_id, type, route, free_timeslot_list, Bw_reqs, TTL, hop_count), where (source, request_id) is used to uniquely identify a packet. The request_id is monotonically increasing, which can be used to detect stale cache route. The type refers to packet type. The route records the path from source to current traversed node. The free_timeslot_list records the status of slot assignment on the route. The hop_count is initially set to zero. The TTL is used to limit the hop count of the path. When a forwarding node receives a non-duplicate RouteReq packet, it checks if there are any common free timeslots between itself and the last node that sends the RouteReq packet. If not, it means that there is no bandwidth to receive from the last node that sends the RouteReq packet. Therefore, the RouteReq packet is dropped. Otherwise, it appends its address, and its free timeslots information to the RouteReq packet and then re-broadcasts the packet. This operation is repeated node by node until the value of TTL is reduced to zero. In order to increase the number of disjoint paths, a forwarding node will re-broadcast a duplicate RouteReq packet that traversed through a different incoming link than the link from which the first RouteReq packet is received, and whose hop count is not larger than that of the first received RouteReq packet.

C. Route Selection and QoS_level Determination

When a destination node receives a RouteReq packet, it checks if there are any common free timeslots between itself and the last node that sends the RouteReq packet. If not, it drops the RouteReq packet. Otherwise, it records this path. If the destination node has a QoS_level one, i.e., it has no more free timeslots, it directly unicasts a RouteRep packet to the multicast source $S$ on the reverse path. Each node on the reverse path receives this RouteRep packet, it marks its timeslots recorded in the RouteRep packet as candidate. This process continues until the RouteRep packet reaches the multicast source $S$. The timeslots status at each node will remain in candidate status until the node receives a TreeConst packet from the multicast source $S$. If no TreeConst packet arrives at the node, the route entry will be deleted and the status of timeslot will be marked as free.

When the destination node has still more free timeslots and it needs more video descriptions, it will not directly unicast the RouteRep packet to the multicast source $S$. It will then wait either for a short time or a reception of a certain number of RouteReq messages. When the destination node receives a proper number of RouteReq messages or after a timeout, it will sort all disjoint paths in descending order according to their number of hops and then selects the proper paths based on shortest path first. After that it sends a RouteRep packet to the multicast source $S$ for each selected paths. The RouteRep packet is treated as mentioned before.

Each destination can determine its QoS_level based on its number of disjoint paths discovered during the route discovery phase. Note that if a destination node has three free timeslots, but only two disjoint paths are discovered then its QoS_level is equal to two. This means that the QoS_level does not depend only on the available bandwidth (i.e., the number of free timeslots) at a destination node, but also it depends on the number of disjoint paths discovered. Fig. 2 shows that a destination node $D$ has two disjoint paths discovered during the route discovery phase. Its free timeslots are $\{ts_1, ts_2, ts_3\}$. The timeslots $ts_2$ and $ts_3$ will be assigned to the links $\{B, R\}$ and $\{A, R\}$, respectively.
As a result, the destination $R$ still has one free timeslot, $t_{s3}$, but it cannot request $QoS\_level$ three because it has only two disjoint paths. Therefore its $QoS\_level$ is equal to two. The $QoS\_level$ for any destination is determined by:

$$QoS\_level(i) = N(P_i)$$

(2)

where $N(P_i)$ is the number of discovered disjoint paths to a destination $R_i$.

### D. Multicast Trees Construction and Video Descriptions Assignment

After a short time, if the multicast source node cannot receive any more $RouteRep$ messages from the destination nodes, the route discovery process completes. At this point, when the route discovery and reply phases are completed, the multicast source records the multiple disjoint paths for each destination $R_i$ in set $P_i$, as seen in Fig. 3(a). After that, it constructs multiple multicast trees according to algorithms 1 and 2. An example of multiple multicast trees construction using algorithms 1 and 2 is shown in Fig. 3. According to Algorithm 1, there are four path sets $P_1$, $P_2$, $P_3$, and $P_4$ from the multicast source $s$ to the destinations $R_{1}, R_{2}, R_{3}$, and $R_{4}$, where:

- $P_1 = \{p_{11}\} = \{S \rightarrow A \rightarrow D \rightarrow G \rightarrow R_1\}$
- $P_2 = \{p_{21}, p_{22}\}$
- $P_2 = \{S \rightarrow B \rightarrow E \rightarrow R_2, S \rightarrow A \rightarrow D \rightarrow G \rightarrow R_2\}$
- $P_3 = \{p_{31}\} = \{S \rightarrow B \rightarrow E \rightarrow R_3\}$
- $P_4 = \{p_{41}, p_{42}\}$
- $P_4 = \{S \rightarrow C \rightarrow F \rightarrow R_4, S \rightarrow B \rightarrow E \rightarrow H \rightarrow R_4\}$

Following step 1, the sets $P_2$, and $P_4$ has the maximum number of paths, which is two, therefore there are two multicast trees according to step 2, namely, $t_1$, and $t_2$. Assume that set $P_2$ is chosen, then the two trees are $t_1 = p_{21}$ and $t_2 = p_{22}$ as seen in Fig. 3(b). The path $p_{11}$ of the destination $R_1$ will be added to $t_2$, according to step 5, since it intersects $t_2$ with the most link. Because $P_1 = \phi$, the algorithm picks up the next destination, $R_3$, and adds its path $p_{31}$ to tree $t_1$ according to step 5. After that, it adds $p_{41}$ to tree $t_2$, and $p_{42}$ to tree $t_1$. The algorithm ends when all the paths of each destination are added. At the end of algorithm 1, two disjoint multicast trees are constructed, namely, $t_1$ and $t_2$ as seen in Fig. 3(d). However, in order to perform the assignment of MD video in a sequential way, the destination nodes on each multicast tree should be superset of the later, i.e., $t_{i+1} \subseteq t_i$. Therefore algorithm 2 is executed to form the final version of the multiple multicast trees. Hence, $R_1$ is added to tree $t_1$. Because Sequential MDMTR maintains totally disjoint multicast trees, the destination $R_2$ is deleted from tree $t_2$. Finally, the multicast trees are constructed, as shown in Fig. 3(e).

After that the source $S$ initiates and sends $TreeConst$ packets to the destination nodes. When a node receives a $TreeConst$ packet, it checks if its address is recorded in the $TreeConst$ packet. If not, the $TreeConst$ is dropped. Otherwise, it marks its timeslots recorded in the $TreeConst$ packet as $reserved$, and records the source address, the multicast group address, and the multicast tree $t$ in the routing table. It then re-broadcasts the $TreeConst$ packet. At the end of this operation, the multicast trees connection is established and the multicast source can begin transmitting video to destination nodes.

### E. Tree Maintenance

As nodes in the network move or as wireless transmission conditions change, some nodes (e.g., forwarders or destination nodes) may become disconnected from the multicast forwarding tree for the group. When a broken link is detected between two nodes on a multicast tree $t$, the two nodes should delete the link from their list of next hops for the multicast group and release all $reserved$ timeslots for this link. The node which is further from the multicast source (i.e., the node downstream of the break) is responsible for initiating the repair of the broken link. The downstream node detects that it has become disconnected from the multicast tree $t$ when it fails to receive a number of successive expected multicast video packets, from its upstream node, on the $reserved$ data timeslot. The downstream node can recognize that it has not received a video packet during a $reserved$ timeslot based on an expected $inter\_arrival$ time for the applications’ packet [8]. Each node maintains a $disconnection\_timer$ for each $reserved$ timeslot. The $disconnection\_timer$ is refreshed each time a packet is received. When a downstream node $F$ detects a break (see Fig. 4), it initiates a $local\_repair$ for the multicast forwarding tree $t$. First, node $F$ sends a $REPAIR\_NOTIFICATION$
packet to the other nodes on the sub-tree (nodes below node $F$) in the multicast distributed tree for source $S$, multicast groupID, and tree $t$. The REPAIR NOTIFICATION packet serves two purposes [8]. It is a notification to nodes in the sub-tree below $F$ that a local repair is in progress and they should not initiate their own local repair. In addition, the REPAIR NOTIFICATION may be received by $F$ parent’s node (node $E$). If node $E$ received the REPAIR NOTIFICATION, it recognizes that one of its child nodes, node $F$, is performing a local repair. The node $E$ then sends a REPAIR NOTIFICATION to node $F$, causing it to cancel its local repair. When a destination node receives a REPAIR NOTIFICATION packet, it postpones its disconnection timer for the amount of time the local repair is expected to take.

If node $F$ has not received a REPAIR NOTIFICATION from its parent node $E$, it initiates a TTL-limited (e.g., $TTL = 2$) ROUTE REPAIR packet with multicast source $S$, multicast groupID, multicast tree $t$, and number of hop_count to the multicast source as a form of network flood, as seen in Fig. 5. A node receiving this ROUTE REPAIR packet can respond if it is a member of the multicast tree $t$, its hop_count to the multicast source is less than or equal to that contained in the ROUTE REPAIR packet, and it has common free timeslots between itself and the last node that sends the ROUTE REPAIR. If the originating node receives more than one REPAIR ACK messages, the node selects the REPAIR ACK packet with minimum hop counts to the multicast source and unicasts a MACT packet to the selected route to activate it. Since the node was repairing a tree break, it is likely that it is now a different distance from the multicast source than it was before the break. If this is the case, it must inform its sub-tree below of their new distance from the multicast source.

If the local repair procedure described above succeeds, the multicast forwarding tree will be reestablished and the destination nodes will continue to receive video data as expected. If the disconnection timer expires at a destination node $R$, this means that the local repair has probably failed. In this case, destination $R$ sends a REJOIN packet to the multicast source with its free timeslots and multicast tree $t$. A node receiving this REJOIN packet can rebroadcast it if it is a member of the multicast tree $t$ and it has common free timeslots between itself and the last node that sends the REJOIN packet. The multicast source may receive multiple REJOIN messages. It will then select the one with shortest path amongst them. After that it sends a MACT packet to the selected path.

**F. Joining a Multicast Group**

When a node wishes to join a multicast group (node $R_3$ in Fig. 6), it initiates a Join Request (JoinReq) packet with
Algorithm 2 Sequential MDMTR: Trees Construction

1: Let $t_1$ be the super multicast tree
2: for $i = 2$ to $L$ do
3: Add the destination nodes to each $t_i$ such that $t_i \subseteq t_{i-1}$
4: end for

Figure 4. Downstream node initiates local repair.

the destination address set to that of the multicast group, with its free timeslots and with hop count equal to zero and broadcasts it to its neighboring node. Any neighboring node (nodes $Y$, $L$, and $X$) receiving this JoinReq packet will rebroadcast it if there are common free timeslots between itself and the node that sends this JoinReq packet. This process will continue until the JoinReq packet reaches the multicast source or a member node (forwarding or destination node on tree $t_1$ or tree $t_2$). When a member node receives this JoinReq packet (nodes $G$, $H$, and $N$), it checks if there is any common free timeslot between itself and the last node that sends the JoinReq packet. If so, there is a path from the multicast source node to the node that initiated the JoinReq packet, $R_1$. After that, a member node (nodes $G$, $H$, and $N$) unicasts a JoinReq packet to its upstream node. This JoinReq packet will re-unicast until it reaches the multicast source $S$. The multicast source may receive multiple JoinReq messages. It then selects the proper disjoint paths and unicasts JoinRep (Join Reply) messages on the reverse paths.

The multicast source will select the shortest path for each multicast tree. For example, it will select the path $S \rightarrow H \rightarrow J \rightarrow K \rightarrow L \rightarrow R_1$, instead of the path $S \rightarrow H \rightarrow M \rightarrow N \rightarrow W \rightarrow Y \rightarrow R_1$, for the first tree ($t_1$) and the path $S \rightarrow E \rightarrow F \rightarrow G \rightarrow X \rightarrow R_1$ for the second tree ($t_2$). Therefore, the destination node $R_1$ will be assigned two video descriptions (its QoS level is equal to two). Fig. 7 shows the structure of multicast trees at the end of joining process.

G. Leaving a Multicast Group

When a leaf destination wishes to leave the multicast group it initiates PRUNE messages and sends them to its upstream nodes and prune itself by deleting all information concerning the multicast group, i.e., source address, multicast group address and releases the reserved timeslots and marks them as free. If a destination is not a leaf node, it cannot leave the multicast group but it can mark itself as a forwarder node. When a node receives a PRUNE packet, it checks in its routing table if it has a downstream node other than the node sending the PRUNE packet. If it has, it cannot prune itself and therefore it should be connected to the tree and it will then drop the PRUNE packet and releases its reserved timeslots between itself and the last downstream node that sent the PRUNE packet (if they are not used for transmission to the other downstream nodes) and marks them as free. Otherwise; it prunes itself and sends the PRUNE packet to its upstream node. Furthermore, it will release its timeslots (for transmission and reception) for this.
session and mark them as free. This process continuous until the existing PRUNE packet arrives at the source node. If a source node receives a PRUNE packet from its downstream node it deletes it from its routing table. After that the source checks if the common timeslots between itself and the deleted downstream node are not reserved with its other downstream nodes. If yes, it releases them and marks them as free. Otherwise; they should be reserved. The process of releasing the reserved timeslots gives the opportunity for other traffics to use these free timeslots.

IV. SIMULATION RESULTS

A. Simulation Framework

We compare the performance of our proposed protocols Sequential MDMTR with that of Serial MDTMR [1]. Serial MDTMR constructs two node-disjoint multicast trees in a distributed manner. The trees are numbered as tree t_1 and tree t_2. Using MDC, Serial MDTMR codes each video frame into two descriptions. Each description is transmitted along one tree. At first, Serial MDTMR build a shortest path multicast tree. Then after requiring all the middle nodes in the first tree not to be middle nodes of the second tree, it constructs another shortest path tree. Since these two trees do not share middle nodes at all, they are node disjoint. Similar to ODMRP, group membership and multicast trees in Serial MDTMR are established and updated by the source on demand.

When a multicast source has packets to send, it periodically triggers a two-step multicast tree construction/refresh process. In the first step, the multicast source broadcasts to the entire network a JoinRequest packet, which includes the treeID. When a node receives a non-duplicate JoinRequest packet for the first tree, it stores the upstream node ID, and rebroadcasts the packet. When the JoinRequest packet reaches a multicast receiver, the receiver unicasts a JoinAck packet to the multicast source via the reverse shortest path. When a middle node in the reverse path receives a non-duplicate JoinAck packet, it updates its corresponding forwarding state in the Forwarding Table, and forwards the packet to its upstream node. Each middle node of the tree only forwards the JoinAck packet once in one tree construction cycle.

After receiving the first JoinAck packet, the multicast source waits for a short time period before broadcasting another round of JoinRequest packet for the second tree in order to ensure the disjointness of two trees. When a node receives a non-duplicate JoinRequest packet, it forwards the packet only if it is not a middle node of the first tree in this round. When the JoinRequest packet reaches a receiver, the receiver unicasts back a JoinAck packet to the multicast source to set up the second tree. It is worth mentioning that Serial MDTMR assigns MD video sequentially to each tree, i.e., it assigns the first description to the first tree, the second description to the second tree and so on. However, the video descriptions, in our proposed protocol Sequential MDMTR, are assigned sequentially to the multicast trees (similar to Serial MDTMR). In contrast to Serial MDTMR, Sequential MDMTR constructs the multicast trees in a centralized manner.

However, Serial MDTMR does not provide QoS capability. Furthermore, it does not take into consideration the heterogeneity of destination nodes. To make a fair comparison, we offer the QoS-extension Serial MDTMR such that each path in the Serial MDTMR protocol adopts Lin’s QoS unicast path routing [7], where MAC sub-layer adopts CDMA-over-TDMA channel model. Furthermore, to enable Serial MDTMR to consider heterogeneous destinations, only destinations that have enough bandwidth will respond to the Join Request messages of the second tree (tree t_2). As a result, all destinations will be connected to tree t_1 while tree t_2 will have only the destinations that are capable of receiving the second description. We use the same topology as shown in Fig. 1. Note, however, that there is only one multicast, and therefore each destination will receive only one description, as seen in Fig. 8. But according to our algorithm there are two multicast trees, see Fig. 3(e).

B. Simulation Scenario

To evaluate performance of the proposed video multicast protocols, extensive simulations have been performed by a simulator written in MATLAB that models a mobile ad hoc network. In the simulation model, the transmission rate is 2 Mbps, and the transmission range is 250 m. In each frame, the data slot in the data phase is set to 5 ms. The total number of slots in the data phase is set to 16. The control slot in control phase is set to 0.1 ms and the total number of slots in the control phase is set to 50. The random waypoint model is used to model the mobility of the nodes [9]. Each node is randomly assigned with an initial location, a destination and a speed. The speed is uniformly distributed between 0 and maximum speed. During the simulation, each node starts its journey from its initial location to the destination at the assigned speed. Upon reaching the destination, another random destination is targeted after a pause time. We only consider the continuous mobility case with zero pause time. The parameters setting for Sequential MDMTR are as follows: Packet inter-arrival time is 100 ms, Start local repair is 100 ms, Local repair TTL is 2, Missing packet to trigger disconnection is 2, Hello message interval is 1 s, and Repair delay is 100 ms. For Serial MDTMR [1] are: JoinRequest interval is 3 s, and Forwarding state lifetime is 4.5 s. Our simulation setup consists of 50 nodes randomly spread in a rectangular terrain of area 1500 × 300 m^2. To change the mobility level of the network, we vary the maximum speed from 3 m/s to 18 m/s. Each simulation runs for a period of 900 s. The results are averaged over 30 simulation runs. The scenarios are generated prior to the simulation so that identical scenarios can be reused for each
case to ensure fairness in the simulation study. One video source and five destinations (each destination node is at least two-hop away from the source) are randomly selected among 50 nodes and recorded so that the same nodes are used for each case to maintain fairness of the comparison study. The raw video is encoded into two descriptions. Each video frame is encoded into two packets using matching pursuits multiple description coding (MP-MDVC) [10] at 65 kbps. The frame rate is set to be 8 fps. We consider interactive video applications in which the playback deadline of each packet is 150 ms after it is generated. If a packet is not received within its playback deadline it is considered lost. In Sequential MDMTR, the number of video descriptions required by a destination node, \( N_{req}(R_i) \), is determined as we explained later in Section III.B. We use the same value of \( N_{req}(R_i) \) for Serial MDTMR protocol. The bandwidth requirement for each description is set to one timeslot.

C. Performance Metrics

We evaluated the performance of Sequential MDMTR and compared it to that of Serial MDTMR using the following metrics:

- **The ratio of user satisfaction:** It is defined as the total number of assigned video descriptions to all destinations divided by the total number of requested video descriptions by all destinations. This measures the efficiency in terms of increasing the number of assigned video description to destination nodes.

- **Number of pure forwarding nodes:** It is defined as the total number of pure forwarding nodes on the multiple multicast trees that are not destinations. This measures the efficiency in terms of minimizing the number of pure forwarding nodes.

- **The ratio of bad frames:** It is defined as the ratio of the number of bad frames experienced in all the destinations to the total number of frames that should have been decoded in all the destinations.

- **The number of bad periods:** A bad period consists of contiguous bad frames. This metric reflects the number of times that received video is interrupted by the bad frames.

- **Normalized packet overhead:** It is defined as the total number of data and control packets generated by the network divided by the total number of data packets actually received. This measures both the data forwarding efficiency and also the control overhead of the multicasting protocol.

- **Control overhead:** It is defined as the total number of control packets generated by the network divided by the total number of successfully decoded video frames at each destination.

D. Varying Number of Multicast Destinations

Fig. 9, Fig. 10, and Fig. 11 illustrate Sequential MDMTR’s, and Serial MDTMR’s performance with varying number of multicast destinations. We use the simulation setup described in Section IV.B with node mobility 3 m/s. The number of destinations was varied from 5 to 26. In Sequential MDMTR, and Serial MDTMR the user satisfaction, as seen in Fig. 9(a), degrades as the number of destinations increases. However, Sequential MDMTR has a higher users satisfaction compared to Serial MDTMR protocol. Fig. 9(b) depicts the number of pure forwarding nodes required to construct and maintain multiple multicast trees. Clearly, the both protocols, almost have the same number of pure forwarding nods.

We can observe from Fig. 10(a) and (b) that Sequential MDMTR outperforms Serial MDTMR in terms of number of bad frames and the number of bad periods. Fig. 11(a) and (b) show that Serial MDTMR has a tremendous overhead since it uses a native flooding based tree construction.

E. Varying Node Speed

Figs. 12 and 13 compare Sequential MDMTR’s, and Serial MDTMR’s performance in different mobility conditions, i.e., varying maximum node speed. The maximum node speed varies from 3 m/s to 18 m/s and the number of destinations is set to 5, a reasonable scalable figure considering the over all population of 50 nodes. Fig. 12(a) and (b)
show the result of the ratio of bad frames and the number of bad periods of the two protocols, respectively. We can observe that Sequential MDMTR protocol achieve a lower ratio of bad frames and the number of bad periods compared to Serial MDTMR protocol. Fig. 13(a) and (b) shows that Sequential MDMTR has lower overhead compared to Serial MDTMR. Again, this is because Serial MDTMR uses a native flooding based tree construction.

V. CONCLUSION

We studied the problem of video multicast for heterogeneous destinations in MANETs. We have proposed multiple disjoint multicast trees with MDC to increase the number
of assigned video descriptions to each destination node. Specifically, we have proposed Sequential MDMTR. Simulation results showed that our proposed protocol outperformed Serial MDTMR in terms of user satisfaction, overhead, the ratio of bad frames, and the number of bad periods.

REFERENCES


