Video Multicast Over Wireless Ad Hoc Networks

Osamah S. Badarneh
Telecommunication Engineering Department
Yarmouk University
Michel Kadoch
Electrical Engineering Department
École de technologie supérieure
{obadarneh@yu.edu.jo, michel.kadoch@etsmtl.ca}

Abstract: - Existing video multicast routing protocols in wireless ad hoc networks have been developed under the assumption that destination nodes wish to receive all the information sent by the multicast source, i.e., they do not support heterogeneous destinations. This paper addresses the problem of video multicast for heterogeneous destinations in wireless 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 three novel multiple multicast trees routing protocols. The first protocol constructs multiple disjoint multicast trees and assigns MD video in a centralized fashion, and is referred to as Centralized MDMTR (Multiple Disjoint Multicast Trees Routing). The second protocol is a variant of Centralized MDMTR. We refer to it as Sequential MDMTR. The main difference between Sequential MDMTR and Centralized MDMTR is that, Sequential MDMTR sequentially assigns MD video to the destination nodes. In order to reduce construction delay and routing overhead, we further propose Distributed MDMTR protocol. Both protocols, Centralized MDMTR and Distributed MDMTR, exploit the independent-description property of MDC along with multiple disjoint paths to increase the number of assigned video descriptions to each destination. We extensively evaluate our proposed protocols by simulations and show that they outperform the existing work.

Key-Words: - Video multicast, wireless ad hoc networks, multiple description coding, heterogeneous destinations

1 Introduction

Wireless ad hoc network is a self-organized and dynamically reconfigurable wireless network without central administration and wired infrastructure. Nodes in a wireless ad hoc network can instantly establish a communication structure while each node moves in an arbitrary manner. Thus a wireless ad hoc network is useful for mobile nodes working in a group to accomplish a certain task. Hence, multicast is very useful and efficient means of supporting group-oriented applications. Multicast is an essential technology for many applications such as video distribution and group video conferencing, data dissemination, disaster relief and battlefield. In recent years, various multicast routing protocols have been proposed for wireless ad hoc networks. The interested readers may refer to Badarneh et. al. [14] for more details.

Video multicasting over wireless ad hoc networks is bandwidth-efficient compared to multiple unicast sessions. However, video multicasting poses great challenges over wireless ad hoc networks [1], [2]. Unlike data packets, video packets are delay and loss sensitive. In addition, due to nodes mobility, the topology of wireless ad hoc networks is frequently changed. As a result, the established links are continuously broken, causing quality loss and interruption in the received video signal. Other challenges include limited battery life of wireless nodes and lower wireless network capacity compared to wired networks.

Research on video streaming over wireless ad hoc networks has gained much attention in recent years [1]–[10]. Most research focuses on multipath video streaming. A popular approach in multipath video streaming is to use a new source coding technique referred to as MDC. The recent advances in MDC made it a promising technology for multimedia applications in wireless ad hoc networks. MDC has been proposed as an alternative of the Layered

Coding (LC) technique. In contrast to LC, MDC fragments a single media stream into independent bit-streams, where the multiple bit-streams are referred to as multiple descriptions, each roughly of equal importance, such that any received description can be independently decoded to give a usable reproduction of the original signal. The quality of the decoded signal is commensurate with the number of received descriptions. The idea of MDC is to provide error resilience to media streams. Since an arbitrary subset of descriptions can be used to decode the original stream, network congestion or packet loss, which is common in best-effort networks such as the Internet, will not interrupt the stream but only cause a temporary loss of quality. The quality of a stream can be expected to be roughly proportional to data rate sustained by the receiver [11].

Video multicast over wireless ad hoc networks has been studied in recent years [1], [7], [8]. 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 node. 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 destinations wish to receive all the video descriptions sent by a multicast source.

In this paper, we exploit the independent-description property of MDC [12], [13] along with multiple trees to propose efficient video multicast routing protocols for heterogeneous destinations. We explain the independent-description property of MDC by the following example. Assume that there are three video descriptions available at the multicast source, namely, MDC1, MDC2, and MDC3 and there are some destination nodes require two video descriptions. Then the multicast source can assign any two descriptions to these destinations, e.g., (MDC1, MDC2), (MDC1, MDC3), and (MDC2, MDC3). The first protocol is called Centralized MDMTR (Multiple Disjoint Multicast Trees Routing). Centralized MDMTR constructs multiple disjoint multicast trees and assigns MD video to a group of destination in a centralized fashion. In contrast, the construction of multiple disjoint multicast trees and the assignment of MD video are done in a distributed fashion in Distributed MDMTR. The third protocol, Sequential MDMTR, is a variant of Centralized MDMTR. Sequential MDMTR sequentially assigns MD video to the destination nodes. This is the main difference that distinguishes it from Centralized MDMTR. Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR protocols aim at increasing the number of assigned video description to destination nodes.

Our contribution is three-fold:

1. We propose novel algorithms for constructing multiple multicast trees and assigning MD video to meet the requirements of destination nodes.
2. We propose efficient multicast routing protocols, for heterogeneous destinations, for video distribution and multicast trees construction.
3. We deploy the independent-description property of MDC to increase the number of video descriptions assigned to each destination node.

The remainder of the paper is organized as follows: First, we review the related work in the next section. Section 3 presents our problem formulation and the network model. In Section 4, we introduce the framework for multiple disjoint multicast trees video streaming. We present the proposed protocols, Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR in Sections 5, 6, and 7, respectively. In Section 8, we evaluate our proposed protocols through simulations, and finally in Section 9 we conclude the paper.

2 Related Work

Video multicast over wireless ad hoc networks with path diversity has been studied in [1], [7]–[10]. Chow and Ishii have proposed a multicast protocol for video transmission called MT-MAODV (Multiple Trees Multicast Ad Hoc On-demand Distance Vector) [8]. An extension to the well-known MAODV to construct two optimally disjoint multicast trees in a single routine for video multicast was proposed. MDC scheme is used to split the video into several independent and equally important video descriptions. Each description is transmitted over different tree.

In [9], the authors introduced a multicast approach for multiple description video over ad hoc networks. An application-centric, cross-layer routing approach with the objective of minimizing the overall video distortion was proposed. In this approach multiple source trees for MD video multicast are used. Furthermore, each description is coded into a base layer and number of enhancement layers. Packets belonging to the same description from both the base layer and enhancement layers are transmitted on the same tree. Authors show that this MD video multicast approach can effectively deal with frequent

link failures and diverse link qualities in wireless ad hoc networks.

Agrawal et al. presented a multiple tree protocol called Robust Demand-driven Video Multicast Routing (RDVMR) [7]. RDVMR explores the path diversity and error resilience properties of MDC. RDVMR deploys a novel path based Steiner tree heuristic to reduce the number of forwarding nodes in each tree, and constructs multiple trees in parallel with a reduced number of common nodes among them to provide robustness against path breaks and to reduces the total data overhead. Two multiple tree multicast routing protocols were presented in [1]. Serial MDTMR protocol (Multiple Disjoint Trees Multicast Routing) constructs two disjoint multicast trees in a serial fashion. However, in order to reduce routing overhead and construction delay of serial MDTMR, parallel MNTMR (Multiple Nearly-disjoint Trees Multicast Routing) was suggested. This protocol constructs two nearly-disjoint multicast trees in a single routine by dividing the network virtually into two parts and tree construction is carried out simultaneously at both virtual topologies. Both serial MDTMR and parallel MNTMR protocols explore MDC to provide robustness for video multicast applications. In order to improve the quality of the received video, the video was split into two descriptions and each description was transmitted over a different tree. The best of our knowledge, there exists no video multicast protocol deploys the independent-description property of MDC.

3 Problem Formulation

3.1 CDMA-over-TDMA MAC Sub-Layer and Timeslot Assignment

In this paper, the MAC sub-layer adopts the well-known CDMA-over-TDMA channel model [17], [18]. The CDMA is overlaid on top of the TDMA infrastructure. Multiple sessions can share a common TDMA slot via CDMA. Observe that, under such a model, the use of a time slot on a link is only dependent on the status of its one-hop neighboring links. Each data phase of a TDMA frame is assumed to be partitioned into K timeslots. A node that wishes to transmit signals must use a free timeslot for transmission, and the node that wishes to receive the signals needs to listen to the transmitting node in the same timeslot. The number of free timeslots over a link represents the free bandwidth on the link. However, the available bandwidth over a link is represented by the number of free timeslots on the link.

In this paper, bandwidth is measured in the unit of free timeslots. It is worth mentioning that, the free bandwidth on the path does not only depend on the free timeslots over the links in the path, but also depend on the timeslot assignment method. Assigning free timeslots to a path to maximize the available bandwidth of the path is NP-hard [18]. In this paper, the timeslot assignment algorithm follows the method proposed in [18]. We assume that each node in the network is equipped with only one transceiver and a node cannot transmit and receive simultaneously at the same time. Furthermore, each node is equipped with a matched filter receiver to detects the transmitted bits. We assume a two-ray propagation model for the wireless channel between any two nodes (power loss ~d^4). We denote the link-gain between node i and node j by H_{ij} . Let P_i denote the power level at which node i transmits. Then node j will receive the information transmitted by node i at a power level P_j , i.e.,

$$P_j = P_i H_{ij} \tag{1}$$

If the information from node i is to be decoded at node j with an acceptable bit error rate, the Signal-to-Interference-plus-Noise Ratio (SINR) at node j must be larger than a given threshold γ, i.e.,

$$SINR_j \geq \gamma \tag{2}$$

where $SINR_j$ is calculated as

$$SINR_j = \frac{P_j}{\sigma^2 + \frac{1}{SF}\sum_{k=1,k\neq j}^{N} P_k H_{kj}} \tag{3}$$

where σ^2 is the noise power, SF is the spreading factor (i.e., processing gain), N is the number of active nodes, P_k is the transmitted power of node k, and H_{kj} is the link-gain between node k and node j.
Using (1), (2), and (3), the minimum transmission power, P_i, required to achieve the SINR target, γ, can be written as:

$$P_i = \frac{\gamma\left(\sigma^2 + \frac{1}{SF}\sum_{k=1,k\neq j}^{N} P_k H_{kj}\right)}{H_{ij}} \tag{4}$$

Distributed Power Control: The iterative power control algorithm where node i with intended receiver node j updates its transmit power via

$$P_i^{(n+1)} = min\{P_i, P_{max}\} \tag{5}$$

where P_i is defined in (4), has been shown to converge to the unique fixed point such that each node i achieves its SINR threshold γ [15]. Note that

each node only needs to know its own received SINR at its designated receiver to update its transmission power. This transmit power update has to be performed by all concurrently transmitting nodes in a distributed fashion. The control channel can be used to facilitate this coordination.

3.2 Problem Statement

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 $\mathbb{R} = \{R_1, R_2, \dots, R_m\}, \mathbb{R} \subseteq V - S$,with corresponding bandwidth requirements $\mathcal{B} = \{b_1, b_2, \dots, b_m\}$.

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, $\mathcal{N}(R_i)$, is maximized, subject to guarantee the bandwidth (B) and SINR requirements (γ). that is

$$\begin{aligned} & maximize\{\mathcal{N}(R_i)\} \\ & s.t. \\ & SNIR_i \geq \gamma \text{ and } BW_i \geq B \end{aligned} \tag{6}$$

This problem is NP-complete in nature and a heuristic approach is used to solve the problem. To resolve this problem, we propose to explore multiple node-disjoint paths along with *the independent-description* property of MDC.

4 Video Multicast Routing Protocols

This paper proposes three on-demand multicast routing protocols for video transmission over wireless ad hoc networks. In the first routing protocol, the assignment of MD video and the construction of multiple disjoint multicast trees are done in a centralized way, for short Centralized MDMTR. In the second routing protocol, the assignment of MD video and the construction of multiple disjoint multicast trees are done in a distributed way. We refer to this protocol as Distributed MDMTR. The third one is a variant of Centralized MDMTR, we refer to as Sequential MDMTR. All protocols aim at increasing the number of assigned video description to each destination node. In order to minimize the packet drop between multiple trees, all protocols construct totally node-disjoint multicast trees.

The multicast source S generates a number of MD video, say $\mathcal{M}$, where each additional description represents a different *QoS_level*. For example, if there are three video descriptions available at the multicast source it represents three possible *QoS_level* (e.g., the first, the second, or the third description represents *QoS_level* one, any two different descriptions represent *QoS_level* two, and three different descriptions represent *QoS_level* three). In this study, we limit the number of *QoS_level* to two, i.e., there are two video descriptions.

4.1 Multicast Packet Forwarding Scheme

Multicast packet forwarding is based on the source S of the multicast packet, multicast *groupId,* and multicast *treeId*. The three protocols, Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR, constructs and maintains totally multiple disjoint multicast trees. Each *tree t* is used to deliver different description of MDC video concurrently. However, the multicast packet forwarding scheme does not support packet forwarding across different trees.

A traffic allocator, at the application layer, is used to split the video traffic. In addition to multicast source S address and the multicast *groupId*, each packet contains a *treeId* field to identify the tree it is meant for. The multicast packet forwarding scheme works as follows. When an intermediate node receives a data packet, it checks its *Membership table* and *Message* cache to avoid forwarding duplicate data packet. the node forwards a non-duplicate data packet, to its downstream node, to a multicast *tree t* if it's a forwarder in that tree, otherwise it drops the packet. This process continues until the packet reaches a leaf destination node.

4.2 Data Structure

- *Multicast routing table*: Each node creates and maintains a Routing Table for the source *S*, and multicast *groupId*. It stores the source address, multicast group *groupID*, the addresses of the upstream and downstream node for the *tree t*, 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 source *S*, and multicast *groupId*. The Membership Table contains the node's status for a particular *tree t*. The status of a node can be any of pure forwarder,

destination, and both forwarder and destination.

- *Message cache*: The message cache is generated and maintained by each node to detect duplicated packets.
- *Timeslot table*: Each node in the network creates and maintains a Timeslot table that contains the status of timeslots (*free, reserved, candidate*). The status of timeslots is exchanged between one-hop neighbor nodes using *Hello* messages.

5 Centralized Multiple Disjoint Multicast Trees Routing Protocol (centralized MDMTR)

Centralized 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 by three-way handshaking approach (Route Request (*RouteReq*), Route Reply (*RouteRep*), and Tree Construction (*TreeConst*) messages).

5.1 Route Discovery

When a multicast source node receives a request from the application layer to set up a QoS multicast connection to a group of destination nodes with bandwidth requirements for each *QoS_level*, it initiates a *RouteReq* message and floods it to its neighbors, as seen in Figure 1(a). The *RouteReq* message 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 message. The *request_id* is monotonically increasing, which can be used to detect stale cache route. The *type* refers to message 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, i.e., nodes *W*, *Z*, *B*, and *C* in Figure 1(a), receives a non-duplicate *RouteReq* message, it checks if there are any common free timeslots between itself and the last node that sends the *RouteReq* message. If not, it means that there is no bandwidth to receive from the last node that sends the *RouteReq* message. Therefore, the *RouteReq* message is dropped. Otherwise, it appends its address, and its free timeslots information to the *RouteReq* message and it then re-broadcasts the message. 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* message that traversed through a different incoming link than the link from which the first *RouteReq* message is received, and whose hop count is not larger than that of the first received *RouteReq* message.

5.2 Route Selection and QoS_Level Determination

When a destination node receives a *RouteReq* message, it checks if there are any common free timeslots between itself and the last node that sends the *RouteReq* message. If not, it drops the *RouteReq* message. 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* message to the multicast source *S* on the reverse path. Each node on the reverse path receives this *RouteRep* message, it marks its timeslots recorded in the *RouteRep* message as *candidate*. This process continues until the *RouteRep* message reaches the multicast source *S*. The timeslot status at each node will remain in *candidate* status until the node receives a *TreeConst* message from the multicast source *S*. If no *TreeConst* message 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 unicasts the *RouteRep* message 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* message to the multicast source *S* for each selected paths. The *RouteRep* message 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. Figure 2 shows that a destination node *R* 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.

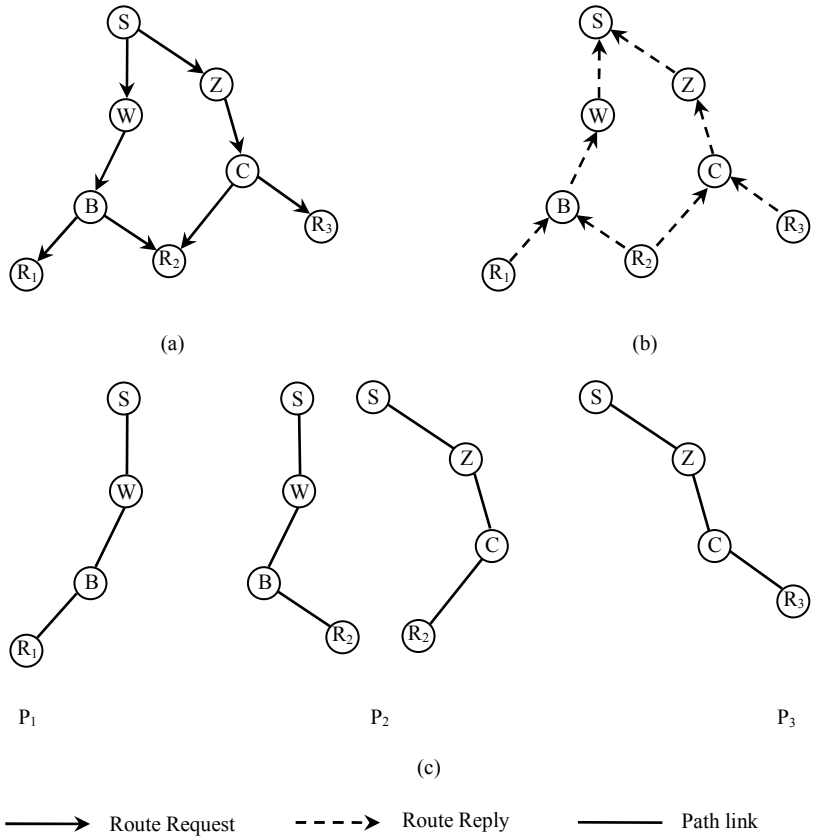


Figure 1. Centralized MDMTR: (a) Broadcasting *RouteReq* message. (b) Uincasting *RouteRep* message. (c) Multiple disjoint paths construction.

As a result, the destination R still has one free timeslots, ts_1, 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:

$$QoS_{level(i)} = \mathcal{N}(P_i) \quad (7)$$

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

5.3 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. After that, it constructs multiple multicast trees according to Algorithm 1. An example of multiple multicast trees construction using Algorithm 1 is shown in Figure 3. According to Algorithm 1, there are three path sets P_1, P_2 and P_3 from the multicast source S to the destinations R_1, R_2 and R_3, where:

$$P_1 = \{p_{11}\} = \{S \to W \to B \to R_1\}$$
$$P_2 = \{p_{21},\ p_{22}\} \to$$
$$P_2 = \{S \to W \to B \to R_2, S \to Z \to C \to R_2\}$$
$$P_3 = \{p_{31}\} = \{S \to Z \to C \to R_3\}$$

Following step 1, the set P_2 has the maximum number of paths, which is two, therefore there are two multicast trees according to step 2, namely, $t_1 = p_{21}$ and $t_2 = p_{22}$ as seen in Figure 3(b). The path p_{11} of the destination R_1 will be added to t_1, according to step 5, since it intersects t_1 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_2 according to step 5. The algorithm ends when all the paths of each destination are added. After that, the source node assigns different video descriptions to each constructed multicast tree. Thus, it assigns the first, second, ..., and L descriptions to *tree* t_1, *tree* t_2, .., and *tree* t_L, respectively. When the source node completes the construction of multicast trees and the

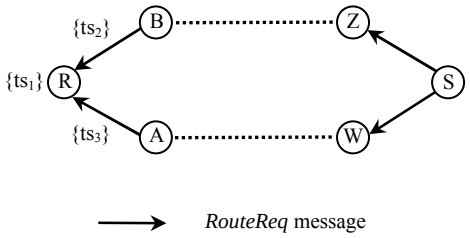


Figure 2. QoS_Level determination.

assignment of video descriptions, it sends this information using *TreeConst* messages to all destination nodes.

When a node receives a *TreeConst* message, it checks if its address is recorded in the *TreeConst* message. If not, the *TreeConst* is dropped. Otherwise, it marks its timeslots recorded in the *TreeConst* message 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* message. At the end of this operation, the multicast trees connection is established and the multicast source can begin transmitting video to destination nodes.

5.4 Multicast Trees Morphing

This phase can be divided into three phases: multiple multicast trees maintenance phase, leaving a multicast group phase, and joining a multicast group phase.

1) *Multiple Multicast Trees Maintenance*: As nodes in the network move or as wireless transmission conditions change, some nodes (e.g., forwarders or destination members) may become disconnected from the multicast forwarding tree of 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 and mark them as *free*. 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 packet from its upstream node on the reserved data timeslot. The downstream node can recognize that it has not received a video packet during the *reserved* data timeslot based on an expected inter-arrival time for the application's packet. The expected packet inter-arrival time may be set to a default value, may be defined according to the port number indicated in the packet, or may be specified by the sending application if an Application Programming Interfaces (API) is available for this purpose [19]. Each forwarder or destination node for the multicast *groupId* and source *S* maintains a *DisconnectionTimer*. The *DisconnectionTimer* is

Algorithm1 Multiple Multicast Trees Construction

1: Find a set P_i of a destination R_i that has the maximum number of paths.
2: Initially, let $T = P_i$, $t_1 = p_{i1}$, $t_2 = p_{i2}$,..., $t_L = p_{iL}$
3: **For** $i = 2$ to m **do**
4: Add each path in P_i to T as follows:
5: Find a path $p_{ij} \in P_i$ such that it intersects a tree, $t_k \subset T$, not covering R_i with the most links, and add p_{ij} to t_k.
6: Remove p_{ij} from P_i.
7: Repeat (5) and (6) until $P_i = \varphi$.
8: **End for**

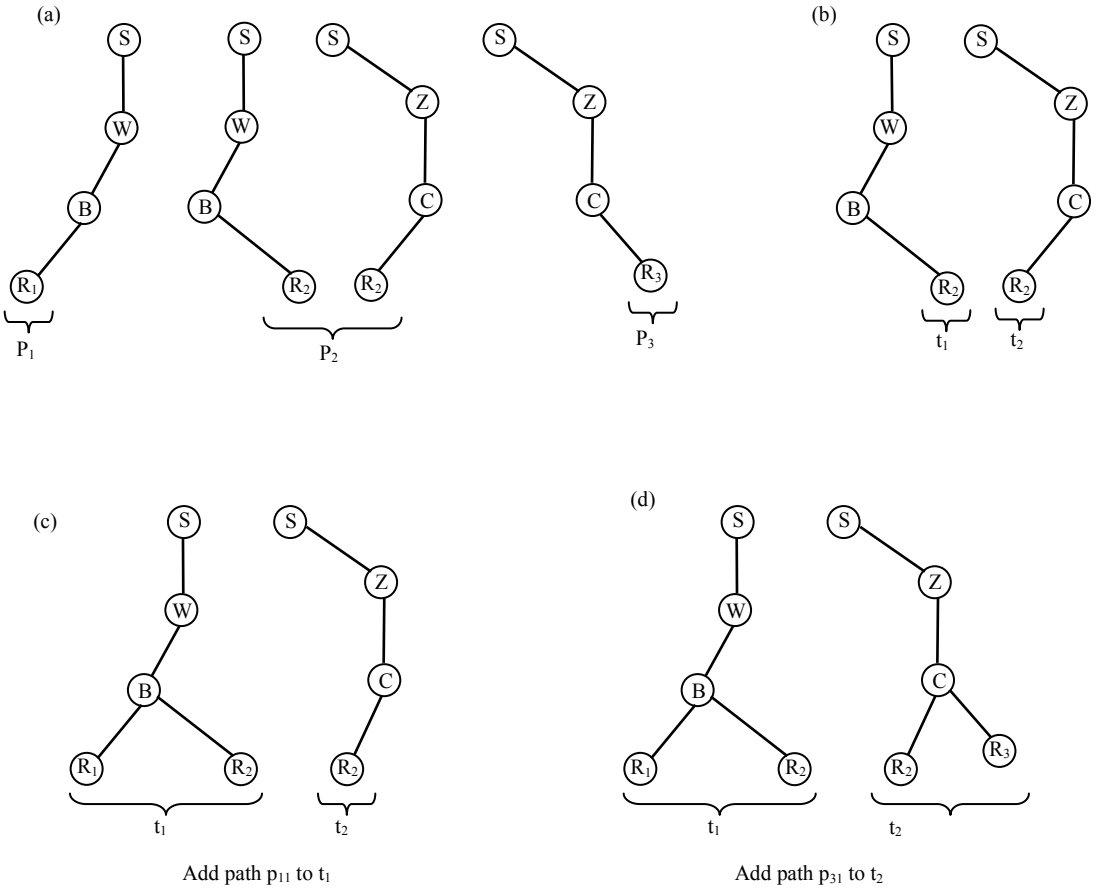


Figure 3. Centralized MDMTR: Multiple multicast trees construction according to algorithm 1.

refreshed each time a video packet is received. The *DisconnectionTimer* is based on the expected packet inter-arrival time value of the last received packet, plus a delay proportional to the node's hop count from the multicast source *S*.

When a downstream node, *K*, detects a break (see Figure 4), it initiates *a local repair* for the multicast forwarding *tree* t_1. At first, node *K* sends a *RepairNotification* message to the other nodes on the sub-tree (nodes below node *K*) in the multicast distributed tree for source *S*, multicast *groupId*, and *tree* t_1. The *RepairNotification* message serves two purposes [19]. It is a notification to nodes in the sub-tree below *K* that a *local repair* is in progress and that they should not initiate their own *local repair*. In addition, the *RepairNotification* message may be received by *K* parent's node (node *J*). If node *J* received the *RepairNotification*, it recognizes that one of its child nodes, node *K*, is performing a *local repair*. The node *J* then sends a *RepairNotification* message to node *K*, causing it to cancel its *local repair*. When a destination node, node R_1 receives a *RepairNotification* message, or when, it initiates *local repair* by sending a *RepairNotification* message, it postpones its *DisconnectionTimer* for a period of time (*Repair Delay*) equivalent to the *local repair* expected time.

After sending the *RepairNotification* message, node *K* waits for a short period of time (*Start Repair*) before it starts its *local repair*. If during *Start Repair* delay node *K* receives a *RepairNotification* message initiated by an upstream node for the same *tree* t_1, multicast *groupId*, and source *S*, then *K* cancels its *local repair*, since the repair should be performed by the downstream node that is adjacent to the broken link.

After *Start Repair* time has expired, and node *K* has not received a *RepairNotification* message initiated by an upstream node, for *tree* t_1, source *S*, and multicast *groupId*, it initiates a TTL-limited (e.g., TTL = 2) *RouteRepair* message with source *S*, multicast *groupId*, *tree* t_1, and the hop count from source *S* to node *K*. This *RouteRepair* message is broadcasted as a form of network flooded.

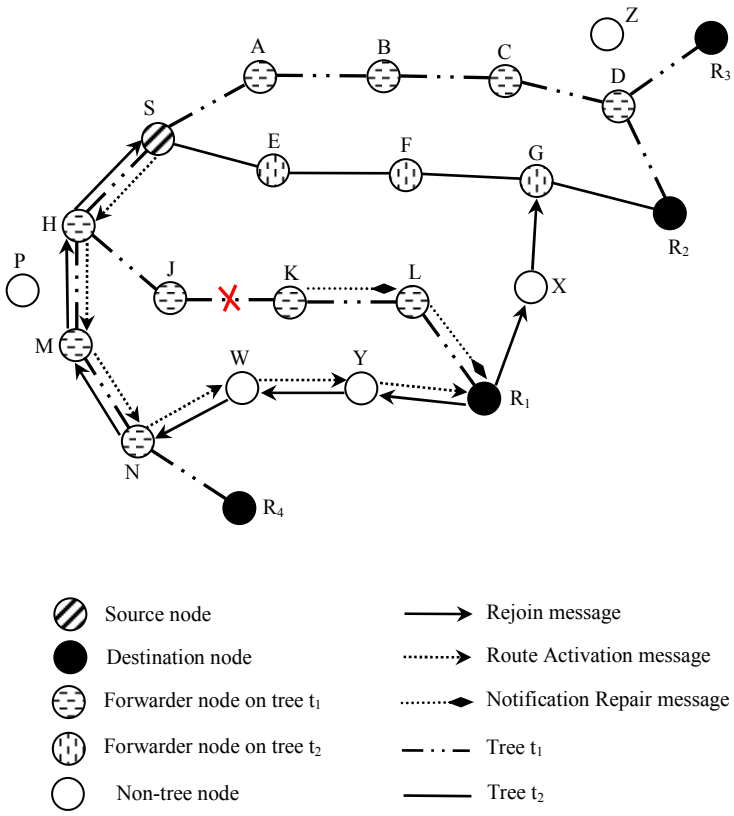


Figure 4. Multicast tree maintenance.

A node receiving this *RouteRepair* message can respond if it is a member of the multicast *tree* t_1, its hop count to the multicast source is less than or equal to that contained in the *RouteRepair* message, and it has common free timeslots between itself and the last node that sends the *RouteRepair* message. If the originating node receives more than one *RepairAck* messages, the node selects the *RepairAck* message with minimum hop counts to the multicast source and unicasts a *RouteActivation* message 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 sub-tree will be reestablished and the destination node will continue to receive multicast video packet as expected. Otherwise, the destination node will rejoin the multicast *tree* t_1 as follows. When the *DisconnectionTime* expires at a destination node R_1, it means that *the local repair* has probably failed. In this case, the destination node, R_1 in Figure 4 should rejoin the multicast *tree* t_1. Node R_1 initiates and broadcasts a *Rejoin* message to the multicast source *S*, multicast *groupId*, and multicast *tree* t_1. Non-member nodes (a node that is not a member of multicast *groupId*, source *S*, and does not belong to *tree* t_1 or *tree* t_2), nodes *X* and *Y* , can rebroadcast the *Rejoin* message if they have common free timeslots with the last node that sends the *Rejoin* message. Non-member nodes will continue to rebroadcast the *Rejoin* message until it reaches the multicast source *S* or a member node, node *N*, (a node that belongs to multicast source *S*, multicast *groupId*, and *tree* t_1).

When a member node, node *N*, receives a *Rejoin* message, it means that there is a path from the multicast source *S* to the destination that initiates the *Rejoin* message (node R_1). After that, node *N*, instead of broadcast the *Rejoin* message, it unicasts the *Rejoin* message to its parent node, node *M*. Finally, the *Rejoin* message will reach the multicast source *S*.

The source *S* may receive multiple *Rejoin* messages. In this case, it will select the one with shortest path and sends a *RouteActivation* message to the destination node R_1. Eventually, the destination node R_1 will receive the *RouteActivation* message and rejoin the multicast *tree* t_1.

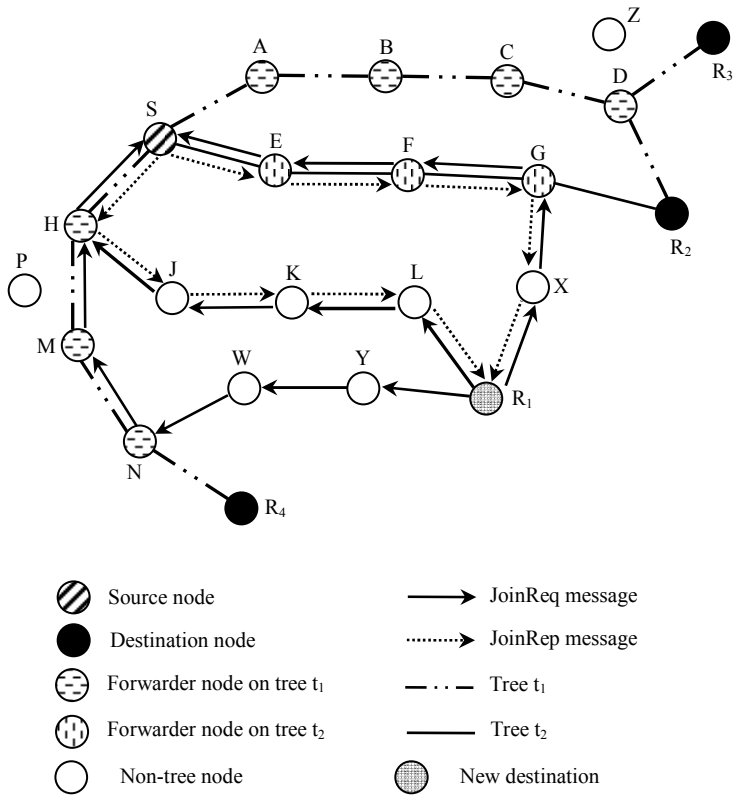


Figure 5. Node joining the multiple multicast trees.

2) *Joining a Multicast Group:* When a new destination node wishes to join a multicast group (node R_1 in Figure 5), it initiates a Join Request (*JoinReq*) message with 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* message will rebroadcast it if there are common free timeslots between itself and the node that sends this *JoinReq* message. This process will continue until the *JoinReq* message reaches the multicast source or a member node (forwarding or/and destination node on *tree* t_1 or *tree* t_2). When a member node receives this *JoinReq* message (nodes G, H, and N), it checks if there is any common free timeslot between itself and the last node that sends the *JoinReq* message. If so, there is a path from the multicast source node to the node that initiated the *JoinReq* message, R_1. After that, a member node (nodes G, H, and N) unicasts a *JoinReq* message to its upstream node. This *JoinReq* message 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). Figure 6 shows the structure of multicast trees at the end of the joining process.

3) *Leaving a Multicast Group:* When a leaf destination node 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. It then 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 forwarding node. When a node receives a *Prune* message, it checks in its routing table if it has a downstream node other than the node sending the *Prune* message.

If it is the case, it cannot prune itself and therefore it stays connected to the tree and then drops the *Prune* message and releases its reserved timeslots (if

they

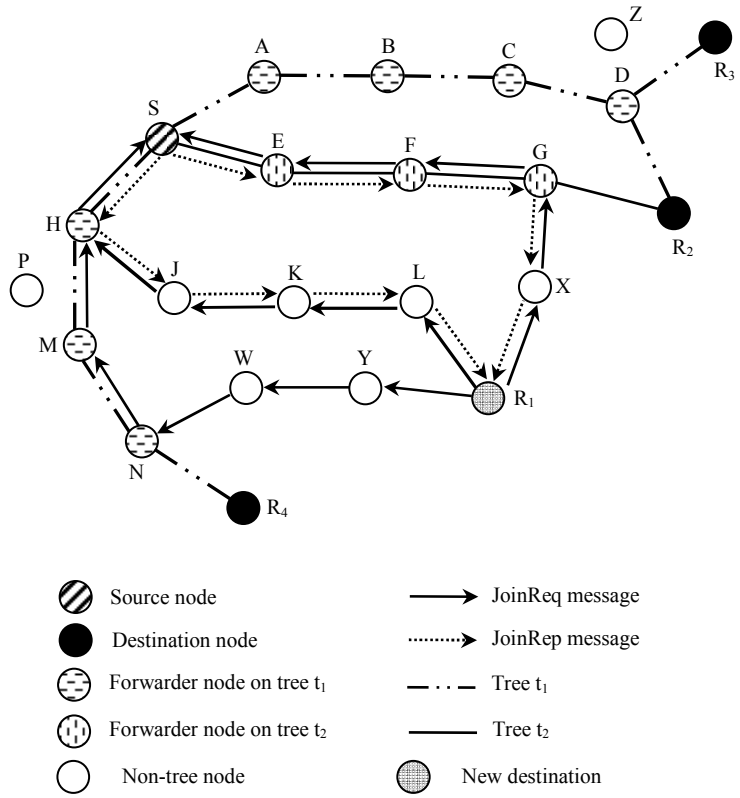


Figure 6. Multiple multicast trees structure after the joining process.

are not used for transmission to the other downstream nodes) for this link by marking them as free. Otherwise; it prunes itself and sends the *Prune* message to its upstream node. Furthermore, it will release its timeslots (for transmission and reception) for this session and will mark them as *free*.

This process continues until the existing *Prune* message arrives at the source node. If a source node receives a *Prune* message from its downstream node it deletes it from its routing table. After that the source checks if the common timeslots with the deleted downstream node are not *reserved* with its other downstream nodes. In that instance, it releases them and marks them as *free*. Otherwise; they stay *reserved*. The process of releasing the *reserved* timeslots gives the opportunity for other traffics to use them.

6 Sequential Multiple Disjoint Multicast Trees Routing Protocol (sequential MDMTR)

Sequential MDMTR constructs multiple disjoint multicast trees and assigns MD video to the destination nodes in a centralized fashion. However, the main difference between sequential MDMTR and centralized MDMTR is that the assignment of MD video is executed in a sequential way. This means that all the destination nodes should be first assigned the first description, then the destination nodes that have *QoS_level two* should be assigned the second description and the destination nodes that have *QoS_level* three should be assigned the third description and so on. Therefore, 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., $t_L \subseteq t_{L-1} \ldots \subseteq t_2 \subseteq t_1$. Algorithm 1 (in Section 5.3) is deployed to construct multiple disjoint multicast trees, and then Algorithm 2 is executed to form the final version of the multiple multicast trees. After that, the trees $t_1, t_2, \ldots, t_L$ will be assigned the first, the second and the L*-th* description, respectively. It is worth mentioning that Serial MDTMR assigns MD video to the destination nodes in a sequential way but in a distributed manner, as we will describe later.

We use Figure 3, to explain how sequential MDMTR constructs multiple disjoint multicast trees. At the end of algorithm 1, two disjoint multicast trees are constructed, namely, t_1 and t_2 as seen in Figure 3(d). However, in order to perform sequential assignment of MD video, R_3 should be connected to t_1. And because Sequential MDMTR maintains

Algorithm 2 Sequential MDMTR: Tree 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 $t_i \subset t_{i-1}$
4: **End for**

totally disjoint multicast trees, therefore, only one multicast is constructed as shown in Figure 7.In Sequential MDMTR, multicast tree maintenance, leaving a multicast group, and joining a multicast group are performed in the same way as in Centralized MDMTR.

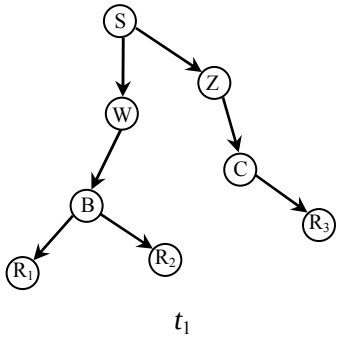


Figure 7. Sequential MDMTR: Multicast tree construction.

7 Distributed Multiple Disjoint Multicast Trees Routing Protocol (Distributed MDMTR)

Distributed MDMTR assigns MD video to the nodes and constructs multiple disjoint multicast trees in a distributed way. The construction of multiple multicast trees and the assignment of MD video are performed by two-way handshaking approach (Route Request (*RouteReq*) and Route Reply (*RouteRep*) messages). Compared with centralized MDMTR, distributed MDMTR offers minimum construction delay and routing overhead. The main difference between Centralized MDMTR and Distributed MDMTR is the assignment of MD video and the construction of multiple disjoint multicast trees. However, in Distributed MDMTR, multicast tree maintenance, leaving a multicast group, and joining a multicast group are performed in the same way as in Centralized MDMTR.

7.1 Multicast Trees Construction and MD Video Assignment

As in Centralized MDMTR, when a multicast source node, in distributed MDMTR, receives a request from the application layer to set up a multicast connection, it broadcasts a *RouteReq* message to its neighbors. The multicast source appends its address, multicast *groupId*, its *free* timeslots, MD video available and the bandwidth requirements for each description in terms of number of timeslots. When a neighbor node receives the *RouteReq* message it checks if there are any common free timeslots between itself and the multicast source, if yes it indicates that this node can be a member of the multicast forwarding group. It then randomly selects one description and rebroadcasts the *RouteReq* message to its neighbors after it appends its address in the *RouteReq* message, *free* timeslots, and the video description that has been selected.

When one of its neighbor nodes receives this *RouteReq* message, it checks if there are any common *free* timeslots between itself and the last node that sends the *RouteReq* message. In the affirmative it rebroadcasts the *RouteReq* message after it appends its address, and its *free* timeslots. Each node that has forwarded the *RouteReq* message should record in its routing table the multicast source address, the multicast *groupId*, and the video description that is recorded in the *RouteReq* message.

In order to maintain totally disjoint multicast trees, each node should rebroadcast only one *RouteReq* message, therefore when a duplicate *RouteReq* message is received the node will drop it.

This process will continue until the *RouteReq* message reaches a destination node. When a destination node receives a *RouteReq* message it checks if there are any common *free* timeslots between itself and the last node that sends the *RouteReq* message, if yes, it records in its routing table the information recorded in the *RouteReq* message. If the destination still has more *free* timeslots this means that it can request more descriptions, therefore it will wait for a short time to select a proper path for each description. After a timeout, the destination unicasts a *RouteRep* message to the multicast source for each selected path.

After a timeout, the multicast source will receive multiple *RouteRep* messages from destination nodes. It will then construct multiple disjoint multicast trees as follows. All nodes that have selected the same description will be on the same tree. Therefore, multiple disjoint multicast trees are constructed. When the multicast source constructs multiple disjoint trees, it starts video transmission to the destination nodes. Figure 8 depicts the assignment of MD video and the construction of multicast tree in Distributed MDMTR.

8 Simulation Results

8.1 Simulation Framework

We compare the performance of our proposed protocols Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR with that of Serial MDTMR [1]. Serial MDTMR, based on ODMRP [16], constructs two node-disjoint multicast trees in a serial 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.

Serial MDTMR constructs two node-disjoint trees in a distributed way. Similar to ODMRP, group membership and multicast trees in Serial MDTMR are established and updated by the source on demand. 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. 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* message, which includes the *treeID*. When a node receives a non-duplicate *JoinRequest* message for the first tree, it stores the upstream node *ID*, and rebroadcasts the message. When the *JoinRequest* message reaches a multicast destination, the destination unicasts a *JoinAck* message to the multicast source via the reverse shortest path. When a middle node in the reverse path receives a non-duplicate *JoinAck* message, it updates its corresponding forwarding state in the Forwarding Table, and forwards the message to its upstream node. Each middle node of the tree only forwards the *JoinAck* message once in one tree construction cycle.

After receiving the first *JoinAck* message, the multicast source waits for a short time period before broadcasting another round of *JoinRequest* message for the second tree in order to ensure the disjointness of two trees. When a node receives a non-duplicate *JoinRequest* message, it forwards the packet only if it is not a middle node of the first tree in this round. When the *JoinRequest* message reaches a destination, the destination unicasts back a *JoinAck* message 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 [17], [18], 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 *JoinRequest* messages of the second tree (*tree* t_2). As a result, all destinations will be connected to *tree* $t1$ while *tree* t_2 will have only the destinations that are capable of receiving the second description. Figure 9 shows how Serial MDTMR constructs multiple multicast trees for heterogeneous destinations.

8.2 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 wireless ad hoc network. In the simulation model, the carrier frequency is 450 MHz and the spreading gain is set to 128. Each node has the maximum transmission power of 33 dBm. The one sided noise PSD is 10^{-9} W/Hz. The SNIR threshold γ is 7 dB. 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 [20]. 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 Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR are shown in Table I and for Serial MDTMR [1] in Table II.

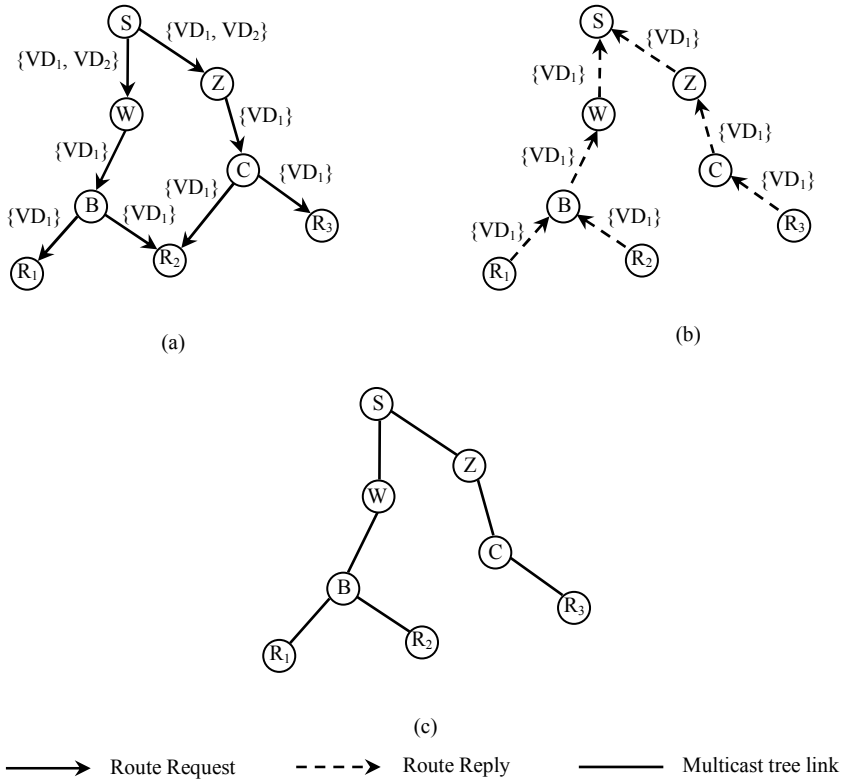


Figure 8. Distributed MDMTR: (a) Broadcasting *RouteReq* message. (b) Uincasting *ReouteRep* message. (c) Multicast tree construction.

Our simulation setup consists of 50 nodes randomly spread in a rectangular terrain of area 1500 X 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) [21] 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 Centralized MDMTR, the number of video descriptions required by a destination node, $\mathcal{N}_{req}(R_i)$, is determined as we explained later in Section 5.2. We use the same value of $\mathcal{N}_{req}(R_i)$ for Sequential MDMTR, Distributed MDMTR, and Serial MDTMR protocols. The bandwidth requirement, *B*, for each description is set to one timeslot.

TABLE I
CENTRALIZED, SEQUENTIAL, AND DISTRIBUTED MDMTR PARAMETERS SETTING

Parameter Name	Value
Packet inter-arrival time	100 ms
Start local repair	100 ms
Local repair TTL	2
Missing packet to trigger disconnection	2
Hello message interval	1 s
Repair delay	100 ms

TABLE II
SERIAL MDTMR PARAMETERS SETTING

Parameter Name	Value
JoinRequest interval	3 s
Forwarding state lifetime	4.5 s

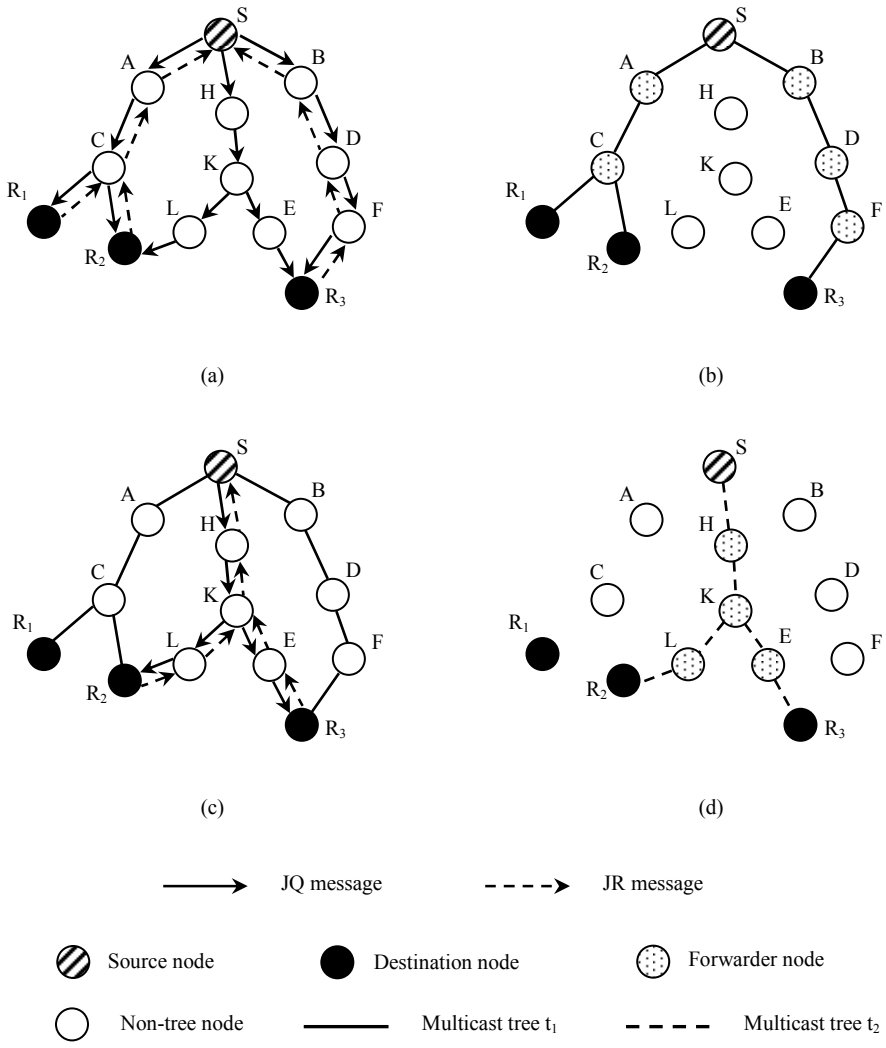


Fig. 9. Multicast trees construction for heterogeneous destinations in Serial MDTMR: (a) First round of *JoinRequest* message. (b) First Multicast tree construction. (c) Second round of *JoinRequest* message. (d) Second Multicast tree construction.

8.3 Performance Metrics

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

- *Multiple Description Assignment Ratio (MDAR):* 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.

$$MDAR = \frac{\sum_{i=1}^{m} \mathcal{N}_{asg}(R_i)}{\sum_{i=1}^{m} \mathcal{N}_{req}(R_i)} \quad (8)$$

where $\mathcal{N}_{asg}(R_i)$and $\mathcal{N}_{req}(R_i)$ represent the number of assigned and requested video descriptions of a destination R_i, respectively.

- *Number of pure forwarding nodes (PFN):* 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.

$$PFN = \sum_{i=1}^{N} X_i \quad (9)$$

where N is the network size and X_i is defined as:

$$X_i = \begin{cases} 1 \; if \; X_i \in T - \{S, \mathbb{R}\} \\ 0 \qquad otherwsie \end{cases} \quad (10)$$

where $T = t_1 \cup t_2$, S is the multicast source, and $\mathbb{R}$ is the multicast group.

- *The ratio of bad frames (RBF):* 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.

$$RBF = \frac{\sum_{i=1}^{m} \mathcal{N}_{bf}(R_i)}{\sum_{i=1}^{m} \mathcal{N}_{f}(R_i)} \quad (8)$$

where $\mathcal{N}_{bf}(R_i)$and $\mathcal{N}_{f}(R_i)$ represent the number of bad frames and the number of frames that should have been decoded of a destination R_i, respectively.

- *The number of bad periods (NBP):* 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 (*NPO): 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 (CO):* 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.

8.4 Varying Number of Multicast Destinations

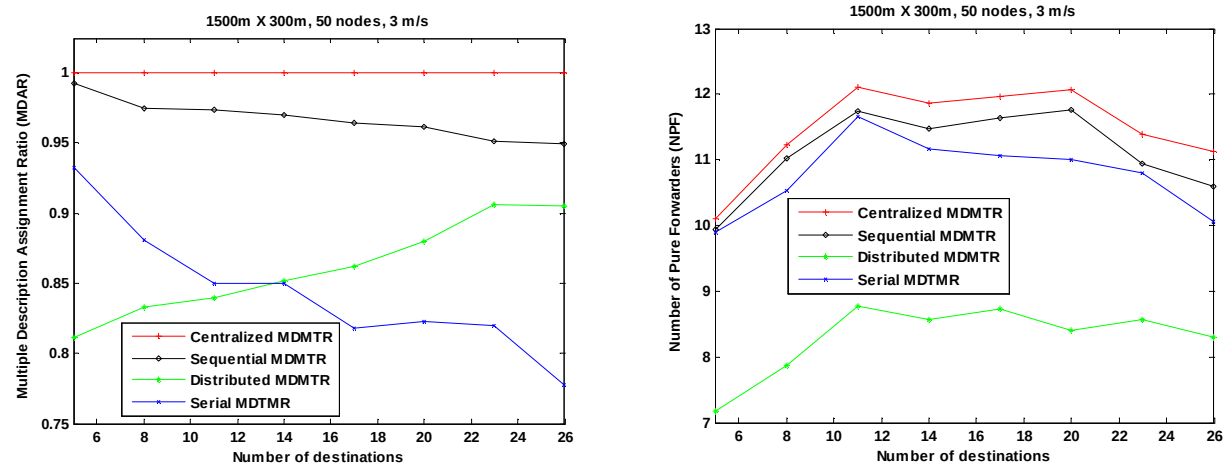
Figures 10, 11, and 12 illustrate Centralized MDMTR's, Sequential MDMTR's, Distributed MDMTR's, and Serial MDTMR's performance with varying number of multicast destinations. We use the simulation setup described in Section 8.2 with node mobility 3 m/s. The number of destinations was varied from 5 to 26. In Sequential MDMTR, and Serial MDTMR the MDAR, as seen in Figure 10(a), degrades with respect to Centralized MDMTR, and Distributed MDMTR. Sequential MDMTR, and Serial MDTMR sequentially assign the video descriptions to each multicast tree, which means that the set of destinations on the previous tree is the superset of the later (*tree* $t_2 \subset$ *tree* t_1). In contrast to Serial MDTMR and Sequential MDMTR, it is not necessarily in Centralized MDMTR and Distributed MDMTR that all the destination nodes should be assigned the first description and then the destination nodes that have a *QoS_level* two should be assigned the second description. Since Centralized MDMTR and Distributed MDMTR deploy the independent-description property of MDC, they achieve higher MDAR than Serial MDTMR. On the other hand, Sequential MDMTR has a higher MDAR compared to Distributed MDMTR. This is because the random assignment of MD video in Distributed MDMTR. The both protocols, Centralized MDMTR and Distributed MDMTR, are well scalable in terms of number of destination nodes as seen in Figure 10(a). When each destination node requests a number of video descriptions that is equal to its number of disjoint paths, Centralized MDMTR will exactly assigns each destination with its requirements. As seen in Figure 10(a) it achieves 100% MDAR. However, Distributed MDMTR deploys the same concept (independent-description property of MDC) when it assigns the video description to the nodes, but it achieves lower MDAR compared to Centralized MDMTR. This is because it randomly assigns the video descriptions to the nodes. As a result, two paths for the same destination node may have the same video description, as seen in Figure 8(a). Figure 10(b) depicts the number of pure forwarding nodes required to construct and maintain multiple multicast trees. Clearly, all the protocols, except Distributed MDMTR, almost have the same number of pure forwarding nods.

We can observe from Figures 11(a) and (b) that Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Serial MDTMR almost achieve the same number of bad frames and the number of bad periods. Serial MDTMR has a tremendous overhead since it uses a native flooding based tree construction. Distributed MDMTR has slightly lower overhead compared to Centralized MDMTR, and Sequential MDMTR since it deploys two-handshaking for constructing multiple multicast trees and assigning MD video.

8.5 Varying Node Speed

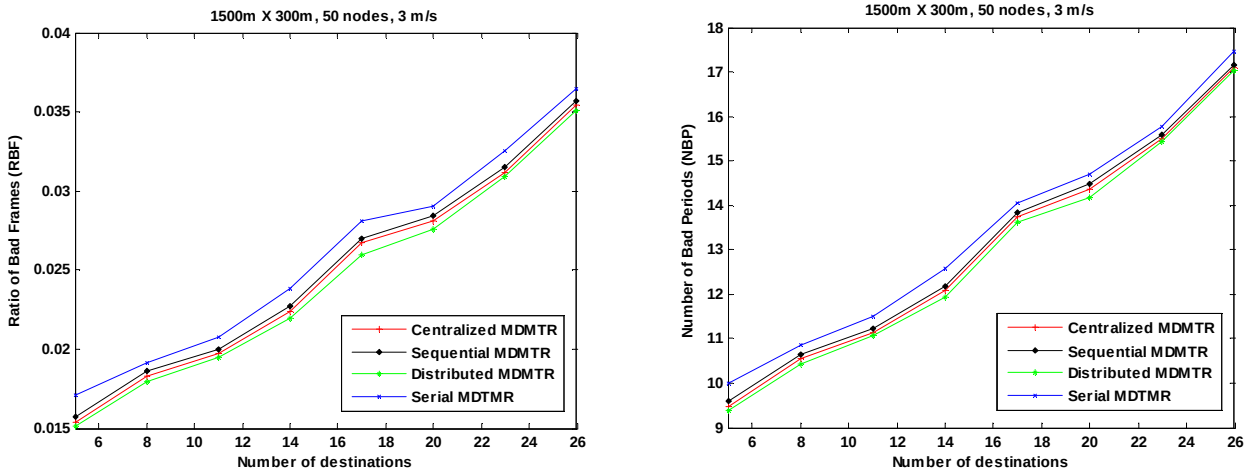
Figures 13 and 14 compare Centralized MDMTR's, Sequential MDMTR's, Distributed MDMTR's, and Serial MDTMR's performance in different mobility conditions, i.e., varying maximum node speed.

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 overall population of 50 nodes.



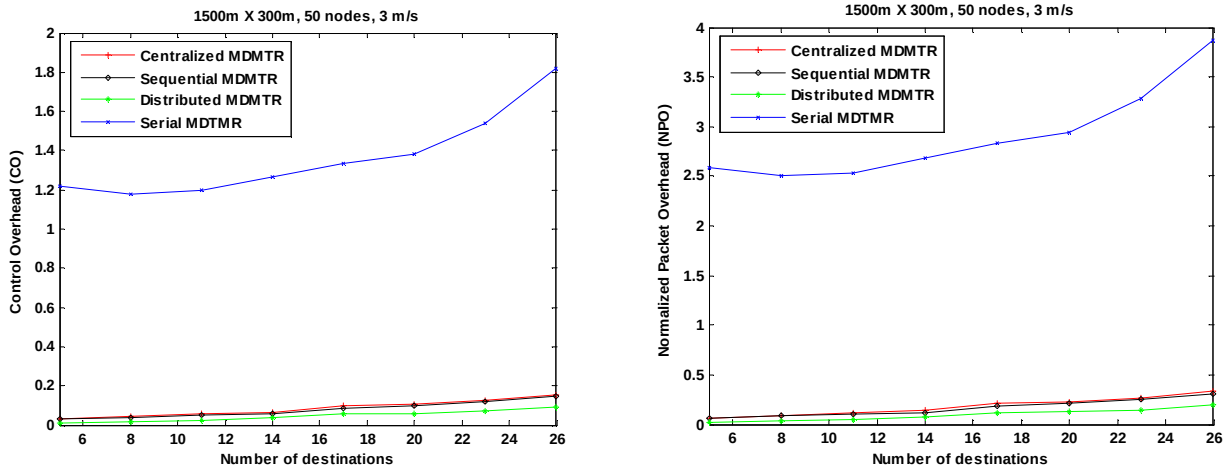
(a) (b)

Figure 10. Varying number of multicast destinations (mobility: 3 m/s): (a) User satisfaction. (b) Number of pure forwarding nodes.



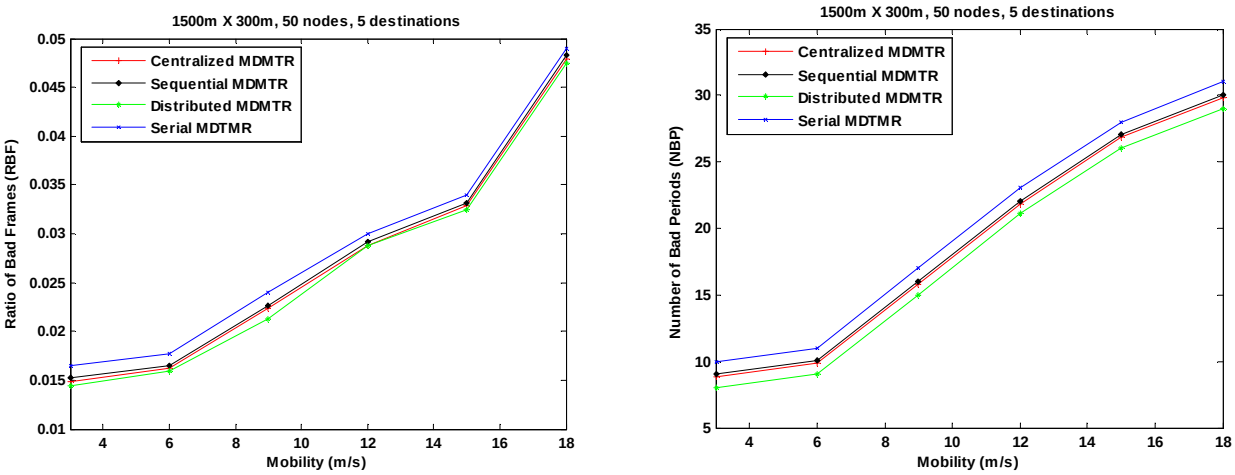
(a) (b)

Figure 11. Varying number of multicast destinations (mobility: 3 m/s): (a) Ratio of bad frames. (b) Number of bad periods.



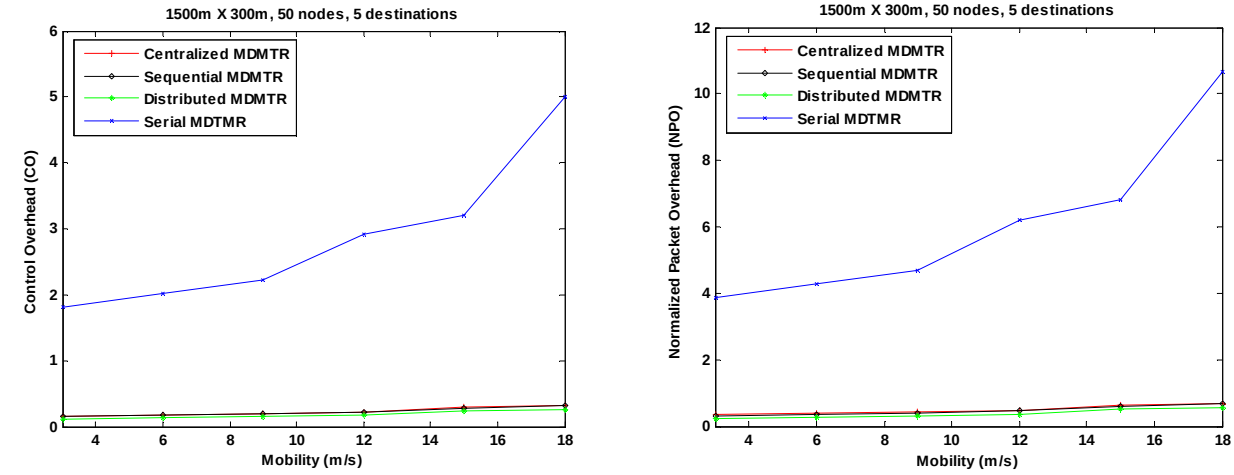
(a) (b)

Figure 12. Varying number of multicast destinations (mobility: 3 m/s): (a) Normalized packet overhead. (b) Control overhead.



(a) (b)

Figure 13. Varying node speed (number of destinations: 5): (a) Ratio of bad frames. (b) Number of bad periods.



(a) (b)

Fig. 14. Varying node speed (number of destinations: 5): (a) Normalized packet overhead. (b) Control overhead.

Figures 13(a) and (b) show the result of the ratio of bad frames and the number of bad periods of the four protocols, respectively. We can observe that both the ratio of bad frames and the number of bad periods are almost the same for all of them. Figures 14(a) and (b) shows that Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR have lower overhead compared to Serial MDTMR. Again, this is because Serial MDTMR uses a native flooding based tree construction.

9 Conclusion

In this paper, we studied the problem of video multicast for heterogeneous destinations in wireless ad hoc networks. We proposed multiple disjoint multicast trees with MDC to increase the number of assigned video descriptions to each destination node. Specifically, we proposed three protocols, namely, Centralized MDMTR, Sequential MDMTR, and Distributed MDMTR. Centralized MDMTR constructs multiple disjoint multicast trees and assigns MD video in a centralized manner. Furthermore, it deploysthe independent-description property of MDC. In Sequential MDMTR, which is a variant of Centralized MDMTR, the assignment of MD video is performed sequentially. On the other hand, Distributed MDMTR constructs multiple disjoint multicast trees and randomly assigns MD video in a distributed fashion. Simulation results showed that our proposed protocols outperformed Serial MDTMR in terms of MDAR, overhead, the ratio of bad frames, and the number of bad periods. Furthermore, they showed good scalability in terms of number of destination nodes.

References:

[1] W. Wei and A. Zakhor, "Multiple tree video multicast over wireless ad hoc networks," IEEE Trans. Circuits Syst. Video Technol., vol. 17, pp. 2-15, 2007.

[2] S. Mao, S. Lin, S. Panwar, Y. Wang, and E. Celebi, "Video transport over ad hoc networks: multistream coding with multipath transport," IEEE J. Sel. Areas Commun., vol. 21, no. 10, pp. 1721-1737, 2003.

[3] S. Mao, D. Bushmitch, S. Narayanan, and S. Panwar, "MRTP: A multiflow real-time transport protocol for ad hoc networks," IEEE Trans. Multimedia, vol. 8, no. 2, pp. 356-369, 2006.

[4] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming in content delivery networks," Proc. IEEE INFOCOM, pp. 1736-1745, 2002.

[5] D. Jurca and P. Frossard, "Video packet selection and scheduling for multipath streaming," IEEE Trans. Multimedia, vol. 9, no. 3, pp. 629-641, 2007.

[6] S. Mao et al., "On joint routing and server selection for multiple description video in wireless ad hoc networks," IEEE Trans. Wireless Commun., vol. 6, no. 1, pp. 338-347, 2007.

[7] D. Agrawal, T. B. Reddy, and C. S. R. Murthy, "Robust demand-driven video multicast over ad hoc wireless networks," Proc. BROADNETS, pp. 1-10, 2006.

[8] C.-O. Chow and H. Ishii, "Multiple tree multicast ad hoc on-demand distance vector (mt-maodv) routing protocol for video multicast over mobile ad hoc networks," IEICE-Trans. Commun., vol. E91-B, pp. 428-436, 2008.

[9] S. Mao, X. Cheng, Y. T. Hou, and H. D. Sherali, "Multiple description video multicast in wireless ad hoc networks," ACM/Kluwer Mobile Networks Appl., vol. 11, pp. 63-73, 2006.

[10] Y. He, I. Lee, and L. Guan, "Optimized Video Multicasting over Wireless Ad Hoc Networks Using Distributed Algorithm," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 796-807, 2009.

[11] V. K. Goyal, "Multiple description coding: Compression meets the network," IEEE Signal Process Mag., vol. 18, pp. 74-94, 2001.

[12] O. Badarneh, *et. all*, "Multiple Description Coding Based Video Multicast over Heterogeneous Wireless Ad Hoc Networks", Proc. IEEE ICC, pp. 1-6, 2009.

[13] O. Badarneh, and M. Kadoch, "An Efficient Video Multicast Routing Protocol for Heterogeneous Destinations in MANETs", to appear in Proceedings of IEEE Symposium on Multimedia, pp.406-411, 2009.

[14] O. Badarneh, and M. Kadoch, "Multicast Routing Protocols in Mobile Ad Hoc Networks: A Comparative Survey and Taxonomy," EURASIP J. Wireless Commun. Networking, vol. 2009, 42 p, 2009.

[15] R. Yates, "A framework for uplink power control in cellular radio systems," IEEE JSAC, vol. 13, pp. 1341-1348, 1995.

[16] S. J. Lee, W. Su, and M. Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," Mobile Networks and Applications, vol. 7, pp. 441-453, 2002.

[17] C.R. Lin, "Admission control in time-slotted multihop mobile networks," IEEE J. Sel. Areas Commun., vol. 19, pp. 1974-1983, 2001.

[18] C.-R. Lin and J.-S. Liu, "QoS Routing in Ad hoc Wireless Networks," IEEE J. Sel. Areas Commun., vol. 17, pp. 1426-1438, 1999.

[19] J.G. Jetcheva, and D. B. Johnson, "Adaptive Demand-Driven Multicast Routing in Multi-hop Wireless Ad Hoc Networks," Proc. ACM MobiHoc, pp. 33-44, 2001.

[20] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," Proc. ACM/IEEE MobiCom, pp. 85-97, 1998.

[21] X. Tang, and A. Zakhor, "Matching pursuits multiple description coding for wireless video," IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 566-575, 2002.

Osamah S. Badarneh received the Ph.D. degree in Electrical Engineering from University of Quebec-École de Technologie Supérieure, ÉTS, (Canada) in 2009. He is currently an Assistant Professor in the Department of Telecommunication Engineering at Yarmouk University (Jordan). From 2001-2006 he was a lecturer at the College of Telecom and Information (K.S.A). He worked in the area of video multicast over Wireless Ad Hoc Network. His current research interests are in designing Medium Access Control and routing protocols/algorithms, Cross-layer design (including the physical layer) and video unicast/multicast routing protocols for Cognitive Radio Networks. He published many papers in international refereed journals and conferences. He is serving as a reviewer for many journals and conferences.

Michel Kadoch (S'67, M'77, SM'04) received the B. Eng from Sir George Williams University (Canada) in 1971, the M. Eng from Carleton (Canada) in 1974, MBA from McGill (Canada) in 1983 and the Ph.D. from Concordia (Canada) in 1991. He is a full professor at École de Technologie Supérieure, ÉTS, (Canada) and the director of the Master Program in engineering. He is active in research mostly in performance analysis and network management and control in wired as well as wireless networks. He is the director of the research laboratory LAGRIT at ETS. He is also an adjunct professor at Concordia University (Canada). He is presently working on Cognitive Radio, Cross layer, and on Reliable multicast in wireless Ad hoc and WiMax networks. Professor Kadoch has published many articles and is the author of a book « Protocoles et reseaux locaux » (Edition ETS, 2004). He is serving as a reviewer for journals and conferences and for grants for NSERC as well as track TPC for ICCCAS, WiMob. He has been involved for many years at ITU-T as a special rapporteur and with the industry namely Teleglobe Canada, CAE, and Communication Canada. He has been a consultant with Harris, Bell South, BC Tel, Concert and British Telecom UK, as well as the CTO (Commonwealth Telecommunication Organization).