Multiple Tree Video Multicast over Wireless Ad Hoc Networks

Wei Wei and Avideh Zakhor, Fellow, IEEE
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, CA 94720, USA
Email: {weiwei, avz}@eecs.berkeley.edu

Abstract

In this paper, we propose multiple tree construction schemes and routing protocols for video streaming over wireless ad hoc networks. The basic idea is to split the video into multiple parts and send each part over a different tree, which are constructed to be disjoint with each other so as to increase robustness to loss and other transmission degradations. Specifically, we propose two novel multiple tree multicast protocols. Our first scheme constructs two disjoint multicast trees in a serial, but distributed fashion, and is referred to as Serial MDTMR. It achieves reasonable tree connectivity while maintaining disjointness of two trees. In order to reduce routing overhead and construction delay, we further propose parallel multiple nearly-disjoint multicast trees protocol, which is also shown to achieve reasonable tree connectivity. Simulations show that resulting video quality for either scheme is significantly higher than that of single tree multicast, with similar routing overhead and forwarding efficiency.

I. Introduction

With the increase in the bandwidth of wireless channels and in the computational power of mobile devices, video applications are expected to become available on wireless ad hoc networks in a near future. Examples of video communication applications over wireless ad hoc networks include spontaneous video conferencing at a location without wireless infrastructure, transmitting video on the battlefield, and search and rescue operations after a disaster.

Video communication is fundamentally different from data communication, since interactive video applications are delay and loss sensitive. Unlike data packets, late arriving video packets are useless to the

video decoder. Thus, the retransmission techniques are not generally applicable to video communication applications with low delay requirements, especially in multicast situations.

There are additional challenges for supporting video communication over wireless ad hoc networks. Due to the mobility of wireless nodes, the topology of ad-hoc networks may change frequently. Thus the established connection routes between senders and receivers are likely to be broken during video transmission, causing interruptions, freezes, or jerkiness in the received video signal. An end-to-end connection route in wireless ad hoc networks generally consists of multiple wireless links, resulting in higher random packet loss than single hop wireless connections in wireless networks with infrastructure, such as base stations. Other challenges include lower wireless network capacity compared to wired networks, and limited battery life of mobile devices. These constraints and challenges, in combination with the delay and loss sensitive nature of interactive video applications, make video communication over wireless ad hoc networks a challenging proposition [2].

Multicast is an essential technology for many applications, such as group video conferencing and video distribution, and results in bandwidth savings as compared to multiple unicast sessions. Due to the inherent broadcast nature of wireless networks, multicast over wireless ad hoc networks can be potentially more efficient than over wired networks [1].

Recently, there has been a great deal of activities on video transport with path diversity, for both wireless ad hoc networks [2-4], and wired networks [5-11]. All of these focus on video unicast applications, and mainly deal with ways to distribute video traffic among multiple paths. A popular approach in multi-path video streaming is to use a new source coding technique referred to as Multiple Description Coding (MDC) [25][26]. MDC is a natural scheme for multiple tree video multicast communication, where feedback-based techniques are not applicable. The basic idea behind MDC is to generate multiple compressed descriptions of the media in such a way that a reasonable reconstruction is achieved if any one of the multiple description is available for decoding, and the reconstruction quality is improved as more descriptions become available [25][26]. The main advantage of MDC over layered coding is that no specific one description is needed in order to render the remaining descriptions useful. As such, unless none of the description make it to the receiver, video quality degrades gracefully with packet loss. However, there is a penalty in coding efficiency and bit rate in using MDC as compared to Single Description Coding (SDC) [25][26]. Specifically, for a given visual quality, the bit rate needed for MDC exceeds that of SDC depending on the number of descriptions.

Multicasting MDC video was first introduced in CoopNet [12] in the context of peer-to-peer networks to prevent web servers from being overwhelmed by large number of requests. CoopNet uses a centralized

tree management scheme, and each tree link is only a logical link, which consists of several physical links and as such is inefficient in wireless ad hoc networks. In [13], the authors propose a genetic algorithm based solution for multiple tree multicast streaming, assuming that (a) they obtain each link's characteristics, and (b) consecutive links' packet loss rates are independent.

There has also been a great deal of prior work in the area of multicast routing in wireless ad hoc networks [14-22]. The On-Demand Multicast Routing (ODMRP) [14] builds multicast mesh by periodically flooding the network with control packets to create and maintain the forwarding state of each node, when the source has packets to send. It takes advantage of the broadcast nature of the wireless network by forwarding group flooding, which provides a certain amount of diversity. A mesh structure is equivalent to a tree structure with *tree flood* enabled[16]. In the remainder of this paper, we refer to ODMRP as a single tree multicast protocol. The Adaptive Demand-Driven Multicast Routing (ADMR) [16] attempts to reduce non-on-demand components within the protocol as much as possible. ADMR does not use periodic network-wide floods of control packets, periodic neighbor sensing, or periodic routing table exchanges. In ADMR, forwarding state is specific to each sender rather than being shared by the entire multicast group. This approach reduces unnecessary forwarding data redundancy. There is also a local subtree repair scheme to detect a broken link by downstream node in ADMR. The Adaptive Core Multicast Routing Protocol (ACMRP) [17] is an on-demand core-based multicast routing protocol that is based on a multicast mesh. A multicast mesh is created and maintained by the periodic flooding of the adaptive core. A core emerges on demand and changes adaptively according to the current network toplogy. This scheme outperforms ODMRP in multi-source scenarios. The Independent-Tree Ad Hoc Multicast Routing (ITAMAR) [21] creates multiple multicast trees based on different metrics in a centralized way. ITAMAR constructs multiple edge disjoint or nearly disjoint trees. The main objective of this protocol is to increase the average time between multicast tree failures. The ITAMAR algorithms are basically based on Dijkstra SPF algorithm [23], which is a centralized approach, and requires knowledge of network topology.

In this paper, we first introduce an architecture for multiple tree video multicast communication over wireless ad hoc networks. The basic idea is to split the video into multiple parts and send each part over a different tree, which are ideally disjoint with each other so as to increase robustness to loss and other transmission degradations. We then propose a simple serial Multiple Disjoint Tree Multicast Routing protocol (Serial MDTMR), which constructs two disjoint multicast trees sequentially in a distributed way, to facilitate multiple tree video multicast. This scheme results in reasonable tree connectivity while maintaining disjointness of two trees.

However Serial MDTMR has a larger routing overhead and construction delay than conventional single tree multicast routing protocols, as it constructs the trees in a sequential manner. To alleviate these drawbacks, we further propose parallel multiple nearly-disjoint trees multicast routing (Parallel MNTMR) in which nearly disjoint trees are constructed in parallel, and in a distributed way. Using the Parallel MNTMR, each receiver is able to always connect to two trees, regardless of the node density. Simulations show that multiple tree video multicast with both Serial MDTMR and Parallel MNTMR improve video quality significantly compared to single tree video multicast; at the same time routing overhead and construction delay of Parallel MNTMR is approximately the same as that of a single tree multicast protocol.

The rest of this paper is organized as follows. In Section II, we introduce the framework for multiple tree multicast streaming. We present the proposed Serial MDTMR and Parallel MNTMR in Sections III and IV respectively. We verify performance of the proposed protocols in Section V. We conclude the paper in Section VI.

II. Multiple tree multicast video communication over wireless ad hoc networks

Our proposed multiple tree multicast video communication system consists of two parts: a multicast routing protocol to construct multiple trees, and a scheme to distribute video packets into different trees. For the latter part, we employ MDC video to form multiple video streams, and transmit different video streams through different trees. In this paper, we assume that the network is lightly loaded, i.e. mobility and poor channel condition rather than congestion are major reasons for packet drop. In this case, multiple tree multicast with MDC effectively alleviates undesirable effects caused by packet drop due to mobility and poor channels.

In this section, we begin by showing the feasibility of multiple tree multicast, and then move on to describe ways to forward packets through multiple trees. We describe the proposed multiple tree protocols in detail in Sections III and IV. Without loss of generality, we limit our discussion to the case of two trees, with each tree carrying one description.

A. Tree Connectivity and Tree Similarity

In order to measure the tree construction capability of multicast routing protocols, we define tree connectivity level P as follows [30]:

$$P \triangleq \frac{E[N]}{M} \tag{1}$$

where M is the product of the total number of receivers and the number of trees, $N = \sum_{i=1}^{m} n_i$, with n_i denoting the number of trees that receiver i connects to, and m denoting the number of receivers. It can be shown that in general $0 \leq p \leq 1$. Given a random topology with n nodes, one random sender and m random receivers, N is the sum of all receivers connected to each multicast tree, and $E[N]$ is the expected value of N over all topologies. Tree connectivity is a measure of the tree construction ability of a multicast routing protocol. For example, if we want to connect one sender to 20 receivers via 2 trees, and the resulting trees connect 18 receivers to 2 trees and 2 receivers to 1 tree, the tree connectivity $P = (18 \times 2 + 2)/(2 \times 20) = 0.95$. Alternatively if only 2 receivers are connected to 2 trees and 18 receivers are connected to 1 tree, the tree connectivity $P = (18 + 2 \times 2)/(2 \times 20) = 0.55$. Obviously, it is desirable to design tree construction scheme with as high tree connectivity level, P, as possible.

To measure the level of disjointness of two trees, we define tree similarity, S, between two trees as the ratio of the number of shared nodes to the number of middle nodes of the tree with a smaller number of middle nodes. Tree similarity between two disjoint trees is zero, and between two identical trees is one. The lower tree similarity between two trees, the lower correlated packet drop across two trees, and hence, the more effective multiple tree video multicasting is in achieving high video quality.

Ideally, we desire a multicast routing protocol to achieve both a high tree connectivity level and a low tree similarity level in video applications space.Intuitively, if the node density is low, it is difficult to construct disjoint trees that connect to all m nodes, and hence either tree connectivity has to be low or tree similarity has to be high. On the other hand, for sufficiently high node density or sufficiently large radio range, we would expect a routing protocol to be able to achieve both a high tree connectivity level and a low tree similarity level.

Thus, an important issue in multiple tree video multicasting is whether or not the required node density to obtain disjoint trees with high connectivity is too high to make it feasible in practice. We have developed the following theorem to address this issue for tree connectivity of two disjoint trees. Before stating the theorem, we need to introduce the term critical density λ_c. Dousse et. al. [31] have stated that there exists one critical density λ_c for a wireless ad hoc network, such that if the density $\lambda < \lambda_c$, all the connected clusters are almost surely bounded; otherwise, almost surely there exists one unique unbounded super connected cluster.

Theorem 1: Consider an infinite wireless network, with nodes assumed to be distributed according to two-dimensional poisson process. Let D_1 denote the required node density to achieve a given tree connectivity level, P, in a single tree case. If $D_1 > \lambda_c$, there exists at least one double disjoint tree

whose required node density D_2 to achieve P satisfies

$$D_2 - \frac{\ln(\pi D_2 r^2 + 1)}{\pi r^2} \leq D_1 \leq D_2 \tag{2}$$

where r is the radio link range.

The detailed proof is included in [30]. We can see from the above theorem that the difference between D_1 and D_2 is only a logarithm factor of D_2, which is small compared to the value of D_1 and D_2. The difference is negligible as $D_1, D_2 \to \infty$, which are requirements for keeping the network connected as the number of total nodes $n \to \infty$ [32][33]. Thus we conclude that the required density for double disjoint tree schemes is not significantly larger than that of single tree schemes, and that tree diversity is a feasible technique to improve the robustness of multicast video transmission over wireless ad hoc networks.

B. Multiple Tree Multicast Packet Forwarding

Our approach is to transmit different descriptions of MDC video flow through different trees simultaneously. If packet drop over two trees are not correlated, when some packets in one tree do not arrive at the destination on time, the receiver continues to decode and display packets corresponding to the other description on the other tree, resulting in acceptable video quality without interruption [24].

Our proposed multiple tree multicast packet forwarding works as follows. The application layer protocol sets a tree-flag in each packet's header to determine the tree to which the packet should be forwarded. The multiple tree multicast protocol forwards the packet in different trees according to the tree-flag as follows: when a node receives a data packet, it checks the node's *Forwarding Table* for the forwarding status and *Message Cache* to avoid forwarding duplicate data packet. The node forwards a non-duplicate packet forwarded in tree-y, if it is a forwarder for tree-y. Each packet flows along the corresponding tree from the sender to the receivers, but is not constrained to follow pre-set branches in the tree, as in the *tree flood* approach [16] or the *forwarding group flooding* approach [14]. Thus our packet forwarding scheme utilizes the broadcast nature of wireless ad hoc networks to obtain extra diversity gain without using extra network resources. Our packet forwarding scheme does not support packet forwarding across the trees, since nodes in one tree are unaware of the status of nodes in the other tree.

III. SERIAL MULTIPLE DISJOINT TREES MULTICAST ROUTING PROTOCOL (SERIAL MDTMR)

Due to the nature of MDC, the less correlated packet drop between two trees, the more robust the video multicast. We assume that the network is lightly loaded, i.e. mobility and poor channel conditions rather

than congestion are major causes of packet drop. In this case, if two trees do not share any middle nodes, packet drop over two trees are independent. Thus our main objective in the design of Serial MDTMR is to construct two node-disjoint multicast trees.

The proposed serial MDTMR constructs two node-disjoint trees in a distributed way. First we 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, we construct another shortest path tree. Since these two trees do not share middle nodes at all, they are node disjoint. Since Serial MDTMR is a way of constructing two disjoint multicast trees, it can be easily applied on top of any suitable single tree multicast routing protocol. Without loss of generality, we design the detailed Serial MDTMR based on ODMRP [14], since ODMRP has been demonstrated to perform well and is well known [15]. By comparing Serial MDTMR and ODMRP, it is easy to quantify the performance gain obtained by the multiple tree multicast routing. We can also design detailed Serial MDTMR based on other multicast routing protocols [14-20], taking advantage of their individual strengths. For example, we could apply a *local repair* scheme similar to [16] to maintain the tree structure with less control overhead. When a middle node or receiver detects that it is disconnected from the corresponding multicast forwarding tree tree-x, where x is 0 or 1, it initiates a *local repair* process for tree-x, which searches the neighborhood of the middle node or receiver in order to find a new upstream node to reconnect the middle node or receiver to tree-x. To keep the disjointness between two trees, the middle node or receiver only selects a node, which is not a forwarding node for tree-(1-x), as its new upstream node.

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 JOIN REQUEST message, which includes the tree ID. When a node receives a non-duplicate JOIN REQUEST message for the first tree, it stores the upstream node ID, and rebroadcasts the packet. When the JOIN REQUEST message reaches a multicast receiver, the receiver unicasts a JOIN ACK message to the multicast source via the reverse shortest path. When a middle node in the reverse path receives a non-duplicate JOIN ACK 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 JOIN ACK message once in one tree construction cycle.

After receiving the first JOIN ACK message, the multicast source waits for a short time period before broadcasting another round of JOIN REQUEST message for the second tree in order to ensure the disjointness of two trees. When a node receives a non-duplicate JOIN REQUEST message, it forwards

the packet only if it is not a middle node of the first tree in this round. When the JOIN REQUEST message reaches a receiver, the receiver unicasts back a JOIN ACK message to the multicast source to set up the second tree.

We compare tree connectivity of single shortest path tree and Serial MDTMR as a function of node density through simulations, as shown in Figure 1. Note that for Serial MDTMR, tree connectivity is averaged across two trees. The total number of nodes is 1000, with 50 receivers. The nodes are randomly distributed according to a two-dimensional poisson process. The results are averaged over 5000 runs. As seen, there is only a small performance gap between the two schemes when node density is larger than 7 nodes per neighborhood. For example, when node density is 8.2 nodes per neighborhood, tree connectivity of a single tree scheme and Serial MDTMR is around 0.99 and 0.95 respectively.

Figure 2 shows simulation results relating the required node density for single tree D_1 and Serial MDTMR D_2 for various tree connectivity levels, ranging from 0 to 1. The value of tree connectivity level for each point can be obtained from Figure 1, using the curve Serial MDTMR and node density D_2. It also shows corresponding lower bounds and upper bounds for node density D_1 according to Equation (2) provided by Theorem 1. As seen, Serial MDTMR curve fits in between the lower and upper bounds quite well, which means that in terms of tree connectivity, the performance of Serial MDTMR is close to that of an ideal double disjoint tree construction scheme.

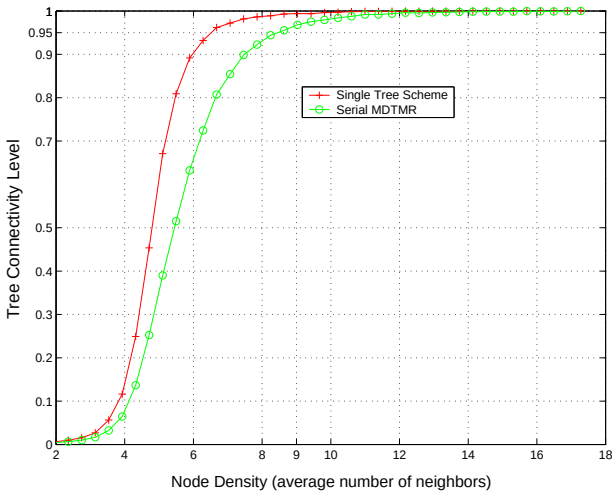


Fig. 1. Tree connectivity of Serial MDTMR

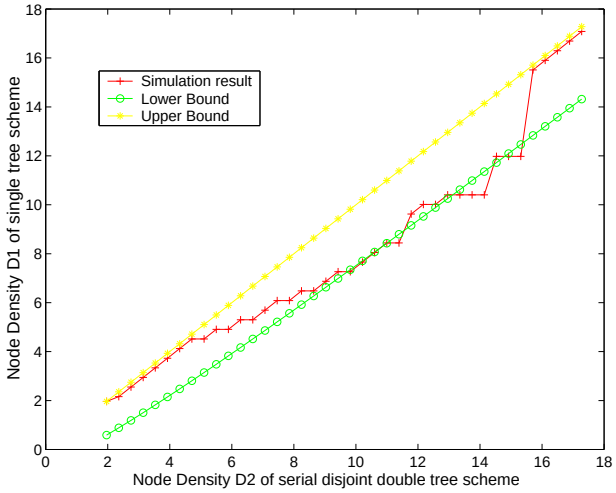


Fig. 2. Comparison of node densities of Serial MDTMR with lower bounds and upper bounds

IV. Parallel Multiple Nearly-Disjoint Trees Multicast Routing Protocol (Parallel MNTMR)

Serial MDTMR achieves reasonable tree connectivity, while maintaining the disjointness of two trees. However its routing overhead and construction delay are potentially twice as much as that of a parallel scheme that would build two trees simultaneously. In this section, we propose a novel parallel double tree scheme, named Parallel MNTMR, to overcome the above disadvantages of Serial MDTMR. We have three main design goals for the Parallel MNTMR:

- *Low routing overhead and construction delay*: The routing overhead and construction delay of Parallel MNTMR should be similar to that of a typical single tree multicast routing protocol.
- *High tree connectivity*: if a receiver is connected to the sender, it should be able to be connected to both trees.
- *Low tree similarity*: The ratio of the number of shared nodes of two trees to the number of nodes of the smaller tree should be minimized.
- *Distributedness*: The protocol should be fully distributed.

If the node density is low, it might not be possible to maintain both a high tree connectivity and a low tree similarity. In this case, Parallel MNTMR would first try to achieve high tree connectivity, and then try to make tree similarity as low as possible. We consider tree connectivity to be more important than tree similarity in the sense that a receiver first needs to connect to the sender in order to receive video

packets.

A. Overview

In a general single-tree multicast protocol, e.g. ODMRP, when a multicast source has packets to send, it triggers a multicast tree construction process by broadcasting a Join-Query (JQ) message to its neighbors. Each node further forwards its earliest received JQ message to its neighbors, until the JQ message arrives at the receivers. Each receiver sends back a Join-Reply (JR) message to the sender to construct the multicast tree.

In Parallel MNTMR, we apply similar JQ and JR processes to construct two nearly-disjoint trees in parallel. We require each node to forward the JQ message at most once in each JQ process, thus the amount of routing overhead of Parallel MNTMR is similar to that of a corresponding single tree multicast protocol.

The basic idea behind parallel tree construction is to classify all the nodes randomly into one of two categories, i.e. group 0 or group 1, based on uniform distribution. The protocol could potentially build tree-0 purely from nodes in group-0, and tree-1 purely from nodes in group-1, resulting in two node-disjoint trees. However this approach achieves low tree connectivity when the node density is not high enough, since it is equivalent to partitioning the network into two networks with half of the node density. Route query messages may sometimes even be blocked in the middle of the network, causing some receivers, which are physically connected to the sender, to be connected to neither tree.

To increase the tree connectivity level, Parallel MNTMR forces each node, which is physically connected to the sender, to forward a JQ message once in a JQ process. We define a *pure JQ message* as one whose route only consists of nodes in the same group, and a *mixed JQ message* as one whose route consists of nodes in both groups. We also classify JQ messages based on the group of the last hop of the message. We call a JQ message a *group-0 message*, if the last hop belongs to group-0; otherwise we call it a *group-1 message*. We summarize the types of JQ messages in Table I.

During the JQ process, nodes store selected JQ messages according to the *JQ message storing condition*, to be described shortly, in their *JQ Message caches*, so that they can use them later for upstream nodes selection in the JR process. A node forwards the earliest received JQ message of the same group as the node belongs to immediately, if it receives one. Otherwise it forwards the earliest received JQ message of the other group, after a short delay d from receiving it. The timeout of the JQ-delay timer should be large enough to differentiate between *pure JQ messages* and *mixed JQ messages*, at the same time as being small enough to reduce the overall delay in constructing the trees. Assuming t_p is the propagation

delay of one hop, the delay of a *pure JQ message* traversing n hops is nt_p. The delay of a n-hop *mixed JQ message* is $nt_p + md$, where m is the number of group node changes within the message, since an extra delay d is incurred every time the current node and the last hop of the message are not in the same group. Thus *pure JQ messages* have lower overall delay, and are therefore selected and forwarded with a priority over *mixed JQ messages*; as will be seen shortly, this improves the disjointness of the constructed trees in the JR process.

After receiving JQ messages, a receiver selects one upstream node for each tree according to the *upstream node selection rule*, to be described shortly, and sends back two Join Reply (JR) messages to the sender via selected upstream nodes to initiate the process of constructing two trees. When a middle node receives a non-duplicate JR message, it also selects its upstream node according to the *upstream node selection rule*, and forwards the JR message to the selected upstream node, until the JR message reaches the sender. The basic idea behind the *upstream node selection rule* is to encourage nodes close to each other select the same upstream node for the same tree, and not to select middle nodes of the other tree. This is because ideally, one would like to make (a) middle nodes of one tree to serve as many nodes in a given neighborhood as possible, and (b) two trees to share as few middle nodes as possible. This is accomplished by synchronizing the selection of upstream nodes for different nodes in a distributed fashion.

TABLE I

CLASSIFICATION OF JQ MESSAGES

Type of JQ messages	Definition
Pure JQ message	A JQ message which is only forwarded by nodes in the same group
Mixed JQ message	A JQ message which is forwarded by nodes in both groups
Group-i JQ message	A JQ message whose last hop is a group-i node

B. Conditions and Rules

Parallel MNTMR applies the following conditions and rules on each node to control the flow of JQ and JR messages, in order to construct two trees with both high tree connectivity and low tree similarity. Without loss of generality, we assume the current node a is in group x, where x could be 0 or 1.

- *JQ message storing condition*: During the JQ process, node a stores received JQ messages in its *JQ Message cache*, so that it can later use them to select its upstream node in the JR process. However if node a stores every received JQ message, it is likely that the protocol constructs trees with loops. For instance, if node b receives a JQ message forwarded by node a, node a is a candidate upstream node for node b; similarly node a could receive a JQ message forwarded by node b, making node b a candidate upstream node for node a. This could potentially result in node a selecting node b and node b selecting node a as their upstream nodes at the same time, thus forming a loop.

  In order to obtain two loop-free trees in the JR process, each node only stores JQ messages satisfying the *storing condition* into its *JQ Message Cache*. Basically the storing condition helps node a eliminate those nodes which are possible offsprings of node a, as candidate nodes for its upstream node, thus avoiding loops in the constructed trees. A JQ message received by node a satisfies the *storing condition*, either if it is the first JQ message that node a receives in the current JOIN-QUERY round, or if the following two conditions are satisfied: (a) the number of hops it has travelled is no larger than that of the first received JQ message of node a plus one, and (b) the JQ message has not been forwarded by node a. Condition (a) helps node a eliminate those nodes, which have a much longer distance from the sender than the shortest distance, as candidate nodes for its upstream node, while condition (b) guarantees the trees to be loop-free.
- *JQ message forwarding condition*: Before node a in group x can decide as to whether or not to forward a JQ message, it has to determine whether a received JQ message is a group-x JQ message. If a JQ message satisfies the *forwarding condition*, it is forwarded immediately. Otherwise, it is either forwarded after a short delay, or not forwarded at all if node a has already forwarded a JQ message in this JQ round. The *forwarding condition* results in *pure group-x JQ messages* being selected and forwarded with a priority over *mixed JQ messages*, thus helping the protocol construct trees that are as disjoint as possible. Formally a JQ message satisfies the *forwarding condition*, if the following two conditions hold true: (a) node a has not forwarded a JQ message in this JOIN-QUERY round, and (b) the message's last hop is the sender or a group-x node.
- *Upstream node selection rule*: The objective of the *upstream node selection rule* is to maximize the disjointness of two trees. Let JQM_a denote the set of all the messages in the *JQ Message Cache* of node a. If there exist both group-0 and group-1 JQ messages in JQM_a, node a selects last hops of the earliest received group-0 and group-1 JQ messages as upstream nodes for tree-0 and tree-1 respectively. Otherwise, we assume all the JQ messages in JQM_a are group-y JQ messages. In this case, if $|JQM_a| > 1$, node a selects last hops of the earliest and the second earliest received JQ

messages as upstream nodes for tree-y and tree-$(1-y)$ respectively; otherwise if JQM_a only has one element, the last hop of the only JQ message is selected as upstream nodes for both tree-0 and tree-1.

Using the *upstream node selection rule*, when nodes select an upstream node for tree-y, they are likely to select the same upstream node for tree-y as other close-by nodes would, and thus avoid the upstream node for tree-$(1-y)$ chosen by other nodes; this increases the likelihood of disjointness of two trees. The *upstream node selection rule* tends to synchronize different nodes' selection. It is effective since neighbors are likely to have received similar JQ messages with similar arrival times. We examine the effect of the *Upstream node selection rule* in the discussion section.

Note that receivers and middle nodes share the same *upstream node selection rule*. Receivers need to apply the rule twice for tree-0 and tree-1 separately, and middle nodes only apply the rule once for the specific tree they belong to.

C. Detailed Double Nearly-Disjoint Tree Construction

Similar to ODMRP, when a multicast source has packets to send, it triggers a multicast tree construction process by broadcasting a JQ message to its neighbors. When a node receives a group-y JQ message, if the message satisfies the *storing condition*, the node stores it into the *JQ Message Cache* for later usage in the JR process; otherwise, the message is simply discarded. If the message also satisfies the *forwarding condition*, the current node forwards the JQ message to its neighbors immediately; otherwise if the JQ message is the earliest received JQ message in the current JQ round, the node sets a JQ-delay timer. When the JQ-delay timer expires, if the node has not forwarded a JQ message in this JQ round, it forwards the earliest received JQ message at that time. The JQ-delay scheme tends to make *pure JQ messages* be selected and forwarded with a priority over *mixed JQ messages* in the distributed tree construction process. We provide flow diagram for processing JQ messages in Figures 3 and 4.

When a receiver receives a group-y JQ Message, if the message is a *pure JQ message*, and the node has not initiated a JR message in this JQ round for tree-y, it selects the last hop of this JQ message as its upstream node for tree-y, and unicasts a JR message to the sender via the selected upstream node, in order to set up tree-y. All nodes, receiving and forwarding the JR message for tree-y, become middle nodes for tree-y. The receiver also sets a timer upon receiving the earliest JQ message. When the timer expires, for each tree for which it has not already initiated a JR message, the receiver selects an upstream node according to the *upstream node selection rule* and unicasts a JR message to the sender via the selected upstream node to construct that tree.

When a middle node receives a non-duplicate JR message for tree-x, it selects an upstream node according to the *upstream node selection rule*, and forwards the JR message to the upstream node. In the end, we obtain one tree mainly consisting of group-0 nodes and another mainly consisting of group-1 nodes. Therefore these two trees are likely to be nearly disjoint.

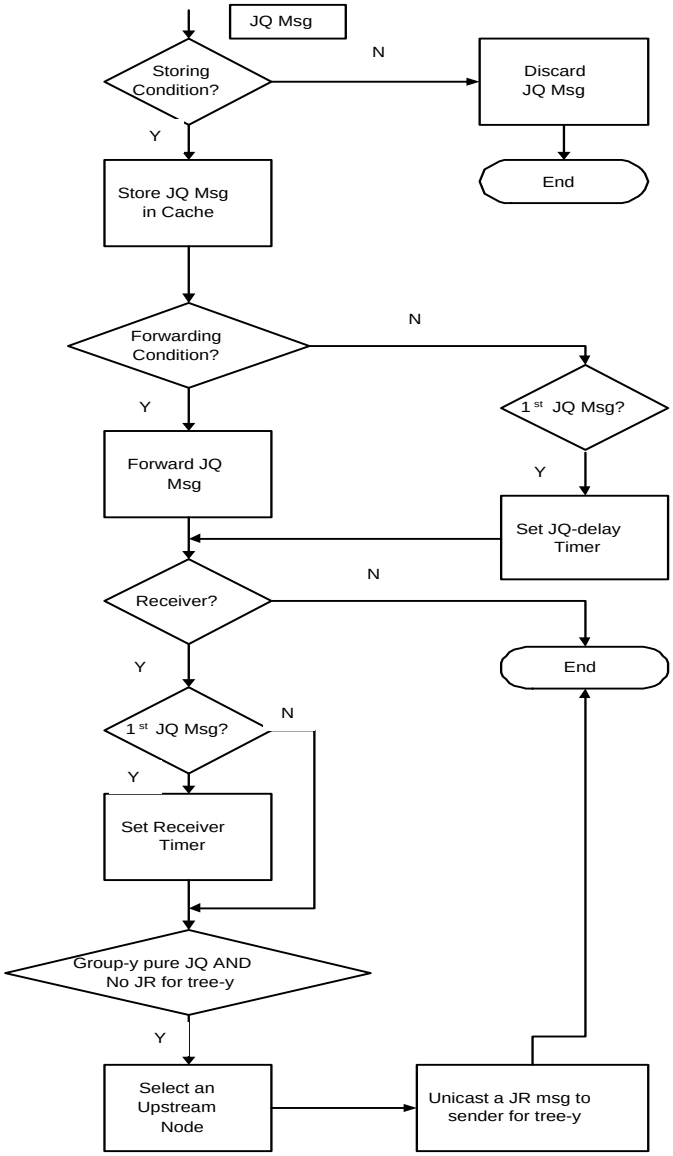


Fig. 3. Flow diagram for processing JQ messages

We visualize the tree construction process in Figure 5. The network consists of one sender (S), two receivers (R1 and R2), and five other nodes. The dashed lines denote the underlying topology of the network, dot-dashed arrows denote the construction of the first tree, and solid arrows denote the

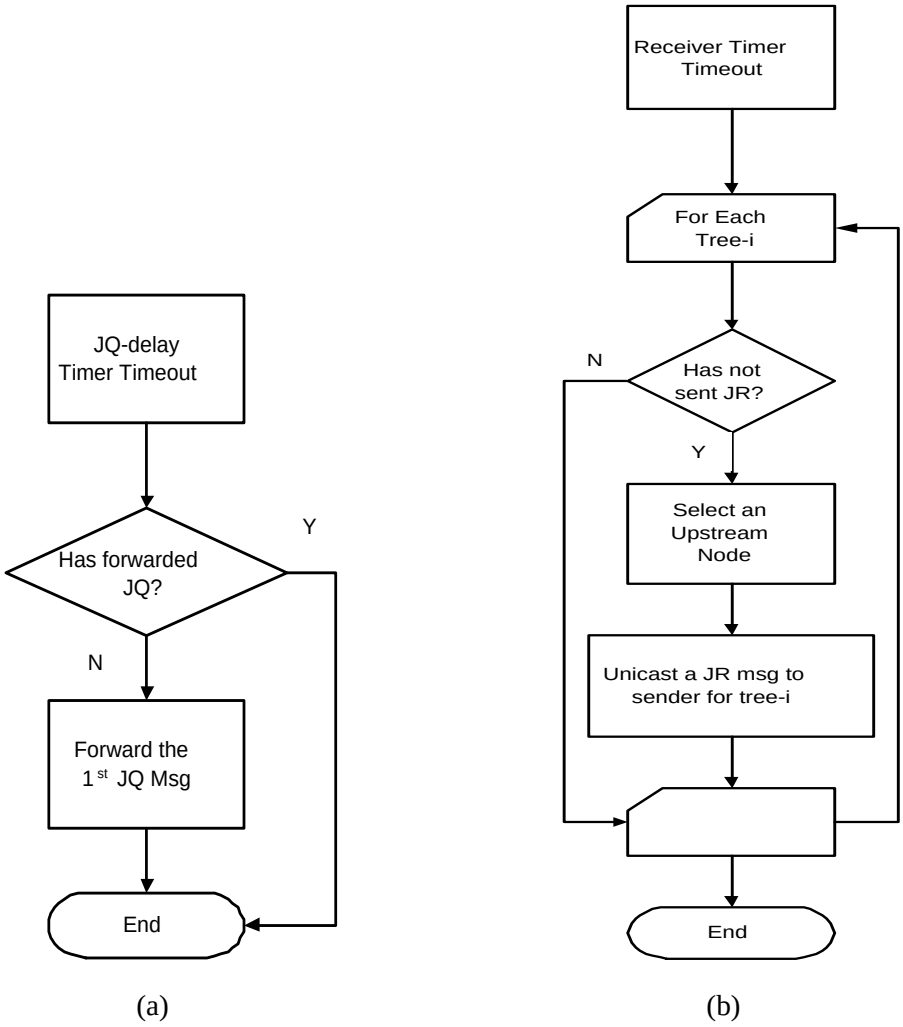


Fig. 4. Auxiliary flow diagrams for processing JQ messages: (a)JQ-delay timer timeout; (b)Receiver Timer timeout.

construction of the second tree. We assume that nodes 1, 3, 4 belong to group-0, and nodes 2 and 5 belong to group-1.

Using our scheme, both R1 and R2 select node 4 as their upstream node for tree-0, and node 5 as their upstream node for tree-1. Let JQ_4 and JQ_5 denote node sets of last hops of JQ messages stored in JQ Messages Caches of nodes 4 and 5 respectively. $JQ_4 = \{1, 2\}$ and $JQ_5 = \{2, 3\}$. According to the *upstream node selection rule*, node 4 selects node 1 as the upstream node for tree-0, since node 1 is a group-0 node, while node 2 is a group-1 node. Using the same rule, node 5 selects node 2 as the upstream node for tree-1. Thus we obtain two disjoint trees, where middle nodes of tree-0 are nodes 1 and 4, and those of tree-1 are nodes 2 and 5. Note that each node learns the group of its candidate

upstream nodes through JQ messages.

In our current implementation, we apply the periodic update scheme of ODMRP [14] to maintain the tree structure, since node movement causes unpredictable tree structure breakage. It is possible to apply other tree maintainence schemes, such as the adaptive demand-driven multicast routing (ADMR) [16], to reduce the amount of control overhead. When a node becomes a forwarder in tree-y, it sets its forwarding status to be 1 for tree-y. Once the forwarding status of a node is updated, it expires after a pre-specified time, i.e. JQ refresh period. The protocol runs JQ and JR processes every JQ refresh period to update forwarding status of forwarders, in order to maintain the tree structure during video transmission.

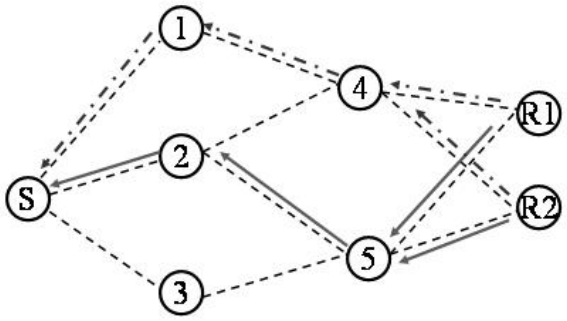


Fig. 5. Double Nearly-Disjoint Tree Construction

D. Discussion

In this section, we argue that the proposed Parallel MNTMR achieves the three design goals we introduced eariler. Firstly, the Parallel MNTMR builds two trees simultaneously, and each node forwards the JQ message at most once in one JOIN-QUERY round, therefore the routing overhead and the construction delay is similar to that of a typical single tree multicast routing protocol. Secondly, as long as a receiver is connected to the sender, the protocol requires it to send JR messages for both trees, therefore the tree connectivity is the same as that of a single tree protocol. Thirdly, regarding the disjointness of the two trees constructed by Parallel MNTMR, we propose the following claim:

Claim 1: Given any two nodes N_a and N_b, which are middle nodes for tree-0 and tree-1 respectively, let JQ_a and JQ_b denote node sets of last hops of JQ messages stored in the JQ Message Caches of nodes N_a and N_b respectively. We sort nodes in JQ_a and JQ_b according to the arrival time of corresponding

JQ messages. Let nodes N_c and N_d denote upstream nodes obtained by the Parallel MNTMR of nodes N_a and N_b respectively. We have $N_c \neq N_d$, if the first two nodes of JQ_a and JQ_b are the same. ■

The proof is shown in the Appendix. Intuitively, *claim 1* shows that if two nodes in different trees share the same first two JQ messages in their JQ message caches, they will not select the same node as their upstream nodes. Thus for many scenarios, the Parallel MNTMR is likely to maintain disjointness between two trees.

We now use the example in Figure 5 to demonstrate that the Parallel MNTMR reduces the number of shared nodes between two trees, in the case that $|JQ_a \cap JQ_b| = 1$. We use a three-bit code to denote the classification of nodes 1, 2, and 3, with the x^{th} bit representing the class to which node x belongs. For example, code 001 shows that nodes 1 and 2 are group-0 nodes and node 3 is a group-1 node. We enumerate all possible classifications of the nodes and all possible arrival sequences at node 4 and 5 of JQ messages forwarded by these three nodes manually, and summarize the results in Table II. From Table II, we see that averaged over all possible classification of all nodes, the probability that two nodes share a upstream node using Parallel MNTMR is $1/6$, while choosing at random would have resulted in $1/4$.

TABLE II

ANALYSIS OF THE SCENARIO SHOWN IN FIGURE 5

Classification of nodes 1, 2 and 3	Probability that node 4 and 5 share the same upstream node
000	1/6
001	0
010	0
011	0
100	3/6
101	0
110	3/6
111	1/6
Average	1/6

V. SIMULATION RESULTS FOR MULTIPLE TREE VIDEO MULTICAST

We compare the performance of our proposed multiple tree multicast communication with that of multicast communication using ODMRP [14] through detailed packet-level simulations in various mobility

and communication scenarios.

A. Simulation Scenario

We use a simulation model based on NS-2 [27]. The Monarch research group in CMU has extended the NS-2 network simulator to include physical layer, link layer and MAC layer models to support multi-hop wireless network simulations [28]. The Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer. The radio model is based on the Lucent/Agere WaveLAN/OriNOCO IEEE 802.11 product, which is a shared-media radio with a transmission rate of 2 Mbps, and a radio range of 250 meters. A detailed description of the simulation environment and the models is available in [28].

The random waypoint model [28] is used to model mobility. Each node starts its journey from a random location to another random destination with a randomly chosen speed, which is uniformly distributed between 0 and maximum speed. Once the destination is reached, another random destination is targeted after a pause. We only consider the continuous mobility case with zero pause time. To change the mobility level of the network, we vary the maximum speed from 2.5 m/s to 15 m/s. For each maximum speed, we randomly generate 30 different scenarios, and average the simulation results over those 30 scenarios.

In each run, we simulate a 50 node wireless ad hoc network within a 1500×300 square meter area. Each simulation is 900 seconds long, and results are averaged over 30 runs. The movement of the nodes and application-layer communication traffic are generated in advance so that they can be replayed identically for different multicast communication protocols.

We randomly choose one sender and eight receivers. For MDC we encode one frame into two packets, while for SDC we encode one frame into one packet. We set the frame rate to 8 fps, and GOP size to 15. For fairness, we set the Peak Signal to Noise Ratio (PSNR) of MDC and SDC to be approximately the same, i.e. 33 dB, where PSNR is a measure of video quality. To achieve approximately same quality, standard MPEG QCIF sequence Foreman is coded with MP-MDVC [25] at 64.9 kbps for MDC, and with Matching Pursuit Codec [29] at 41.2 kbps for SDC. We use the coded video traces to set size of video packets of the simulations. We consider interactive video applications in which the playback deadline of each packet is 150 milliseconds (ms) after it is generated.

B. Performance Metrics

To describe the metrics we use, we first introduce the definition of a bad frame.

Definition 1: A description of an I-frame is decodable at the playback deadline, if the packet corresponding to the description is received. A description of a P-frame is decodable, if at the playback deadline, both the packet corresponding to the description is received and the same description of the previous frame is decodable. A frame of a MDC stream is called a bad frame, if neither one of its two descriptions is decodable; a frame of a SDC stream is called a bad frame, if it is not decodable. For MDC I frames are repeated in each description.

We evaluate the performance of the multiple tree multicast communication with Parallel MNTMR using the following metrics:

a. **The ratio of bad frames:** In multicast scenario, the ratio of bad frames is the ratio of the number of bad frames experienced in all the receivers to the total number of frames that should have been decoded in all the receivers. Note that the ratio of bad frames is a different metric from packet delivery ratio or the number of packet loss bursts due to the following two reasons. First it considers the correlation between different frames. For example, for SDC, if an I-frame is a bad frame, all the following P-frames in the same group of pictures (GOP) are considered to be bad frames, regardless of whether or not packets representing those P-frames are received. Second this metric reflects the fact that MDC can to some extent conceal the undesirable effects caused by missing packets, by decoding and displaying one of the two descriptions.
b. **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.
c. **Normalized packet overhead:** The total number of control packets transmitted by any node in the network, divided by the total number of video frames received by all the receivers. This metric represents the control packet overhead of the routing protocol normalized by the successful video frames received.
d. **Forwarding efficiency:** The total number of data packets transmitted by any node in the network, divided by the total number of packets received by all the receivers. This metric represents the efficiency of multicast forwarding of the routing protocol. For video applications, forwarding efficiency is more important than the control packet overhead, since the size of a video packet is generally much larger than the size of a control packet.
e. **Average hops of each packet:** The average number of hops that each packet takes. This metric represents the quality of the multicast trees as constructed by the multicast routing protocol.
f. **Tree similarity:** The ratio of the number of shared nodes to the number of middle nodes of the tree

with a smaller number of middle nodes. This metric shows the level of disjointness of two trees.

C. Simulation Results

We compare the following four schemes:

- Multiple tree video multicast with Parallel MNTMR and MDC;
- Multiple tree video multicast with Serial MDTMR and MDC;
- Single tree video multicast with ODMRP [14] and MDC;
- Single tree video multicast with ODMRP and SDC.

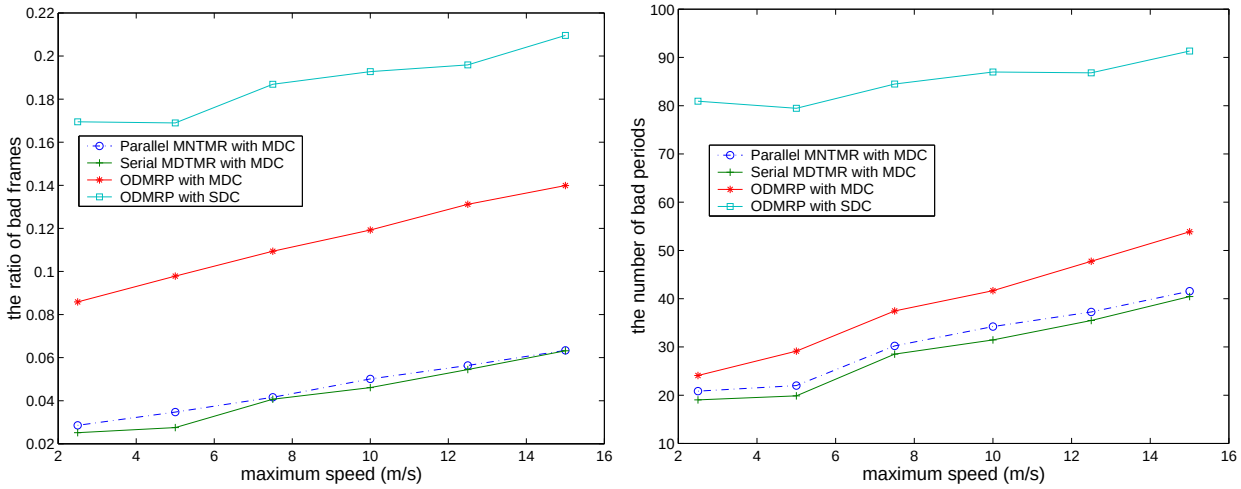
The reason we choose ODMRP as a benchmark for single tree protocols is that it outperforms many single tree protocols [15]. Through forwarding group flooding technique, ODMRP uses redundant links inside the mesh structure to forward data packets, thus increasing packet delivery ratio. In our simulations, we use ODMRP implemented by J. Jetcheva [16], who realized all specifications in the original paper [14].

ITAMAR is another multiple tree protocol, which builds edge disjoint or nearly-disjoint trees. However there are two obvious advantages of our proposed techniques as compared to ITAMAR[21]: first, our protocols are distributed, rather than centralized, and hence do not require the knowledge of network topology in advance; second, our protocols' overhead is $O(n)$, rather than $O(n^2)$ of ITAMAR[21], where n is the number of total nodes.

For fair comparison, all of three multicast routing protocols use 3 seconds for the JOIN REQUEST flooding interval, and use 4.5 seconds as a forwarding state lifetime.

Figures 6(a) and 6(b) show the result of the ratio of bad frames and the number of bad periods of the four schemes respectively. As expected, both the number of bad frames and the number of bad periods increase with maximum speed. As seen, performance of multiple tree multicast with Parallel MNTMR is close to Serial MDTMR, and they both perform much better than the other two schemes with ODMRP. Shown in Figure 7, two trees obtained by Parallel MNTMR only share approximately eight percent of nodes, which means they are nearly disjoint. This explains the reason the two multiple tree protocols perform similarly. The combination of our proposed multiple tree multicast protocols, e.g. Parallel MNTMR or Serial MDTMR, and MDC reduces contiguous packet loss caused by broken links of multicast tree, since links of two nearly-disjoint trees fail nearly independent, resulting in much better received video performance than that with ODMRP and MDC. By comparing ODMRP with MDC and ODMRP with SDC respectively, we conclude MDC by itself could also reduce scattered packet loss

caused by wireless channel error, or packets collision, thus reducing both the ratio of bad frames and the number of bad periods.



(a) (b)

Fig. 6. Performance evaluation for multiple tree video multicast: (a) The ratio of bad frames; (b)The number of bad periods.

We plot PSNR and loss traces of a randomly selected receiver using Parallel MNTMR with MDC and ODMRP with SDC in Figures 8 and 9 respectively. Every node moves randomly with a maximum speed 5.0 m/s. For MDC, it can be seen in Figure 8(a) that PSNR drops gracefully, when there is packet loss only in one substream. As seen in Figure 8, in Parallel MNTMR, most of the time, packet losses of two substreams do not overlap, thus reducing both the number and the amount of PSNR drops. The PSNR curve of ODMRP with SDC shown in Figure 9(a) has more frequent and severe drops than that of Parallel MNTMR with MDC; this is because PSNR drops for every packet drop in SDC video, and would drop severely when there is a burst of packet loss. We also visually examine the reconstructed video sequences under different schemes. For the video sequence obtained via Parallel MNTMR with MDC, we experience 6 short periods of distorted video in 900 seconds, while for the video sequence obtained via ODMRP with SDC, we experience 16 longer periods of more severely distorted video in the same time period.

Figure 10(a) shows the normalized control packets for the four schemes. Simulation results show that the number of normalized control packets of Parallel MNTMR is very similar to that of ODMRP, and is about 50 percent lower than that of Serial MDTMR. In order to construct double disjoint trees, Serial MDTMR has to broadcast JQ message twice in each routing cycle, while both Parallel MNTMR and

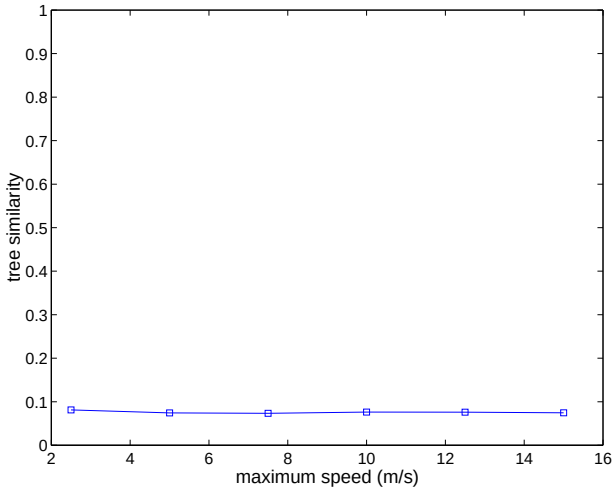


Fig. 7. Tree Similarity of Parallel MNTMR

ODMRP only broadcast once. Let n and m denote the number of nodes and receivers respectively, and k_1 and k_2 denote the number of middle nodes in tree-0 and tree-1 respectively. The number of control messages in one Join Query round of ODMRP, Parallel MNTMR and Serial MDTMR are $n + m + k1$, $n + (m + k1) + (m + k2)$, and $2n + (m + k1) + (m + k2)$ respectively. In general multicast scenarios, $(m + k_i) \ll n$, $i = 0, 1$, or else simple broadcasting scheme is more efficient than multicasting. Thus we see that Parallel MNTMR has approximately the same control overhead as ODMRP, and they both have significantly lower overhead than Serial MDTMR.

Figure 10(b) shows that the number of the normalized forwarded data packets is almost the same for all four schemes with Parallel MNTMR being slightly worse. This indicates that the performance gain of Parallel MNTMR and Serial MDTMR is not at the expense of forwarding a packet more times than ODMRP, rather by the combined effect of independent trees and MDC. Shown in Figure 11, the average number of hops travelled by each packet using Parallel MNTMR is similar to Serial MDTMR, and is only approximately four percent higher than that using ODMRP.

Figures 12(a) and 12(b) compare Serial MDTMR with MDC, Parallel MNTMR with MDC, ODMRP with MDC and ODMRP with SDC as a function of cross traffic level in the network. The maximum speed of each node is 5.0 m/s. There are four UDP flows in the network, connecting randomly selected senders and receivers. Each flow has the same rate, i.e. one fourth of the total flow rate. With the increase of cross traffic rates, the possibility of packet drop due to collision and congestion increases, thus both the ratio

of bad frames and the number of bad periods of all four protocols increase. As seen, the performance of Serial MDTMR and Parallel MNTMR with MDC is better than ODMRP with MDC and ODMRP with SDC under varying cross traffic levels. From simulation results shown in Figure 12, our schemes work well, when the network is lightly loaded or medium loaded. When the network is heavily loaded, none of existing protocols would work. In that scenario, we could potentially combine our protocols with some multicast rate control schemes to reduce the network load.

Figure 13 shows the comparison of four schemes as a function of the number of receivers ranging from five to eighteen. Two multiple tree streaming schemes outperform ODMRP with SDC and ODMRP with MDC for different number of receivers.

We also compare four schemes as a function of node density of the network, and show the results in Figures 14(a) and 14(b). We fix the size of the area to be 1000×1000 square meters, and vary the node density from 2.5 to 9.0 neighbors per node by varying the number of nodes in the network. The maximum speed of each node is 5.0 m/s. As seen, the performance of two multiple tree streaming schemes is better than that of ODMRP with SDC for all node densities. When the node density is smaller than 6 neighbors per node, the performance of Parallel MNTMR with MDC and Serial MDTMR with MDC improve with node density, since our proposed multiple tree protocols build more disjoint trees with higher node density. However when the node density is higher than 6 neighbors per node, the performance of two multiple tree streaming schemes gets worse with higher node density, because of the increase in routing overhead caused by larger number of nodes in the network. From Figure 14(c), the control overhead of Serial MDTMR grows faster than that of Parallel MNTMR with the increase of node density, which explains the reason that Parallel MNTMR outperforms Serial MDTMR when node density is higher than 6 neighbors per node.

We also show packet loss rate, average delay and the number of middle nodes of each tree in Figure 15. Shown in Figure 15(c), for Serial MDTMR the average number of middle nodes of tree 1 is around 13 percent higher than that of tree 2, and for Parallel MNTMR, the average number of middle nodes of tree 1 is around 4 percent lower than that of tree 2. Although two trees obtained from Serial MDTMR is a little unbalanced, shown in Figures 15(a) and (b), packet loss rate and average delay of tree 1 are very similar to those of tree 2 in different simulation scenarios. Specifically, for Serial MDTMR, average packet loss rate and delay of tree 1 is only 6 and 1 percent lower than those of tree 2.

VI. CONCLUSIONS

In this paper, we study the problem of real-time video multicast communication over wireless ad hoc networks. We have proposed multiple tree video multicast with MDC to provide robustness for video multicast applications. Specifically, we first propose a simple distributed protocol, Serial MDTMR, which builds two disjoint trees in a serial fashion. This scheme results in good tree connectivity while maintaining disjointness of two trees. In order to reduce the routing overhead and construction delay of Serial MDTMR, we further propose Parallel MNTMR, which constructs two nearly disjoint trees simultaneously in a distributed way. Simulation shows that video quality of multiple tree multicast video communication is significantly higher than that of single tree multicast video communication, with similar routing overhead and forwarding efficiency.

APPENDIX

Proof of *Claim 1*:

We prove the claim through enumerating all possible scenarios. We list all the scenarios in Table III.

TABLE III

ALL SCENARIOS OF CLAIM 1

Scenario Number	Types of Nodes in JQ_a	Types of Nodes in JQ_b
1	All group-0 nodes	All group-0 nodes
2	All group-0 nodes	Both group-0 and group-1 nodes
3	All group-1 nodes	All group-1 nodes
4	All group-1 nodes	Both group-0 and group-1 nodes
5	Both group-0 and group-1 nodes	All group-0 nodes
6	Both group-0 and group-1 nodes	All group-1 nodes
7	Both group-0 and group-1 nodes	Both group-0 and group-1 nodes

Let message sets JQM_a and JQM_b denote JQ Message Caches of nodes N_a and N_b respectively. In scenarios 2, 6 and 7, according to the *upstream node selection rule*, N_c is the last hop of the first received group-0 JQ message in JQM_a, and N_d is the last hop of the first received group-1 JQ message in JQM_b. Therefore $N_c \neq N_d$.

In scenario 1, using the *upstream node selection rule*, N_c is the last hop of the first received group-0 JQ message, which is also the first received JQ message in JQM_a. N_d is the last hop of the second

received JQ message in JQM_b. Since the first two JQ messages of JQM_a and JQM_b are the same, $N_c \neq N_d$. Similarly in scenario 3, we arrive at the same conclusion.

In scenario 4, N_c is the last hop of the second received JQ message. Since the first two JQ messages of JQM_a and JQM_b are the same, the first two JQ messages of JQM_b are group-1 JQ messages. Thus N_d is the last hop of the first group-1 JQ message, which is also the first JQ message. Therefore $N_c \neq N_d$. We could arrive at the same conclusion in scenario 5 in a similar fashion.

Therefore for all seven possible scenarios, $N_c \neq N_d$.

REFERENCES

[1] K. Obraczka, and G. Tsuduk. "Multicast routing issues in ad hoc networks", *IEEE International Conference on Universal Personal Communications*, 1998.

[2] S. Mao, S. Lin, S. Panwar, and etc. "video transport over ad hoc networks: multistream coding with multipath transport", *IEEE Journal on Selected Areas in Communications*, Dec. 2003, pp. 1721 - 1737.

[3] S. Mao, S. Lin, S. Panwar, and Y. Wang. "reliable transmission of video over ad-hoc networks using automatic repeat request and multi-path transport", *Proceedings of the 2001 Fall IEEE Vehicular Technology Conference*, Atlantic City, NJ, October 2001, pp.615-619.

[4] S. Lin, Y. Wang and S. Panwar. "video transport over ad-hoc networks using multiple paths", *invited paper in the Proceedings of the 2002 IEEE International Symposium on Circuits and Systems*, Scottsdale, Arizona, May 2002, pp. 57-60.

[5] J. Apostolopoulos. "Reliable video communication over lossy packet networks using multiple state encoding and path diversity", Visual Communications and Image Processing (VCIP) 2001.

[6] T. Nguyen and A. Zakhor. "Multiple Sender Distributed Video Streaming", *IEEE Transactions on Multimedia*, Vol. 6, No. 2, April 2004, pp. 315 - 326.

[7] T. Nguyen and A. Zakhor. "distributed video streaming over internet", *in Multimedia Computing and Networking 2002, Proceedings of SPIE*, San Jose, California, January 2002, Vol. 4673, pp. 186-195.

[8] T. Nguyen, and A. Zakhor. "distributed video streaming with forward error correction", *Packet Video 2002*, Pittsburgh, April 2002.

[9] J. Apostolopoulos, W. Tan, S.J. Wee, G.W. Wornell. "Modeling Path Diversity for Multiple Description Video Communication", *IEEE Inter. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2002.

[10] Y. J. Liang, E. Setton, and B. Girod, "Channel-Adaptive Video Streaming Using Packet Path Diversity and Rate-Distortion Optimized Reference Picture Selection," *Proceedings IEEE Fifth Workshop on Multimedia Signal Processing*, St. Thomas, Virgin Island, Dec. 2002.

[11] J. H. Kim, R. M. Mersereau, and Y. Altunbasak, "Network-adaptive video streaming using multiple description coding and path diversity," *in Proc. of IEEE Intl. Conf. on Multimedia and Expo (ICME)*, Baltimore, MD, July 2003.

[12] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. "Distributing streaming media content using cooperative networking," *ACM NOSSDAV*, Miami Beach, FL, USA, May 2002.

[13] S. Mao, X. Cheng, Y. Hou, H. Sherali, "Multiple description video multicast in wireless ad hoc networks," in Proc. BroadNets 2004, Oct. 25-29, San Jose, CA.

[14] S. Lee, M. Gerla, and C. Chiang. “On-Demand Multicast Routing Protocol”, *Proceedings of IEEE WCNC’99*, New Orleans, LA, Sep. 1999, pp. 1298-1302.

[15] S. Lee, W. Su, et al. “A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols”, *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[16] J. Jetcheva and D. Johnson. “Adaptive Demand-Driven Multicast Routing in Multi-hop Wireless Ad Hoc Networks”, ACM Mobihoc 2001.

[17] S. Park, and D. Park. “Adaptive core multicast routing protocol”, *Wireless Networks 2004.*

[18] C. Toh, G. Guichala, and S. Bunchua. “ABAM: OnDemand Associativity-Based Multicast Routing for Ad Hoc Mobile Networks”, *In Proceedings of IEEE Vehicular Technology Conference, VTC 2000*, pages 987–993, September 2000.

[19] J. Xie, R. Talpade, et al. “AMRoute: ad hoc multicast routing protocol”, *Mobile Networks and Applications*, v.7 n.6, p.429-439, December 2002.

[20] C. Wu and Y. Tay. “AMRIS: A multicast protocol for ad hoc wireless networks”, *in Military Communications Conference Proceedings. 1999*, vol. pp. 25-29. New York: IEEE, 1999.

[21] S. Sajama, and Z. Haas. “Independent-tree ad hoc multicast routing (ITAMAR)”, *ACM Mobile Networks and Applications*, Oct. 2003.

[22] C. Cordeiro, H. Gossain, D, Agrawal. “Multicast over wireless mobile ad hoc networks: present and future directions”, *IEEE Network*, Jan./Feb. 2003.

[23] E. W. Dijkstra. “A note on two problems in connexion with graphs”, *Numerische Mathematik*, 1 (1959), pp. 269-271.

[24] W. Wei and A. Zakhor, “Multipath Unicast and Multicast Video Communication over Wireless Ad Hoc Networks”, *International Conference on Broadband Networks (Broadnets) 2004*, San Jose, CA, USA, Oct. 2004 *(invited)*, pp. 496 - 505.

[25] X. Tang, and A. Zakhor. “Matching pursuits multiple description coding for wireless video,” *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 566 -575, June 2002.

[26] Y. Wang, and S. Lin. “Error-resilient video coding using multiple description motion compensation,” *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 438-452, June 2002.

[27] ns-2: network simulator. http://www.isi.edu/nsnam/ns/

[28] J. Broch, D.A. Maltz, D.B. Johnson, et al. “A performance comparison of multi-hop wireless ad hoc network routing protocols”, *ACM MobiCom 1998*, pp.85-97.

[29] R. Neff and A. Zakhor. “Very Low Bit-Rate Video Coding based on Matching Pursuits”, *IEEE Transactions on Circuits and Systems for Video Technology*, February 1997, vol. 7, no. 1, pp. 158-171.

[30] W. Wei and A. Zakhor, “Connectivity for Multiple Multicast Trees in Ad Hoc Networks”, in International Workshop on Wireless Ad-hoc Networks (IWWAN), Oulu, Finland, June 2004.

[31] O. Dousse, P. Thiran and M. Hasler, “Connectivity in ad-hoc and hybrid networks”, *IEEE Infocom 2002*

[32] T. Philips, S. Panwar, A. Tantawi. “Connectivity Properties of a Packet Radio Network Model”, *IEEE Trans. On Information Theory*, Sep. 1989

[33] P. Gupta, and P. Kumar. “Critical Power for Asymptotic Connectivity in Wireless Networks”. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming* 1998

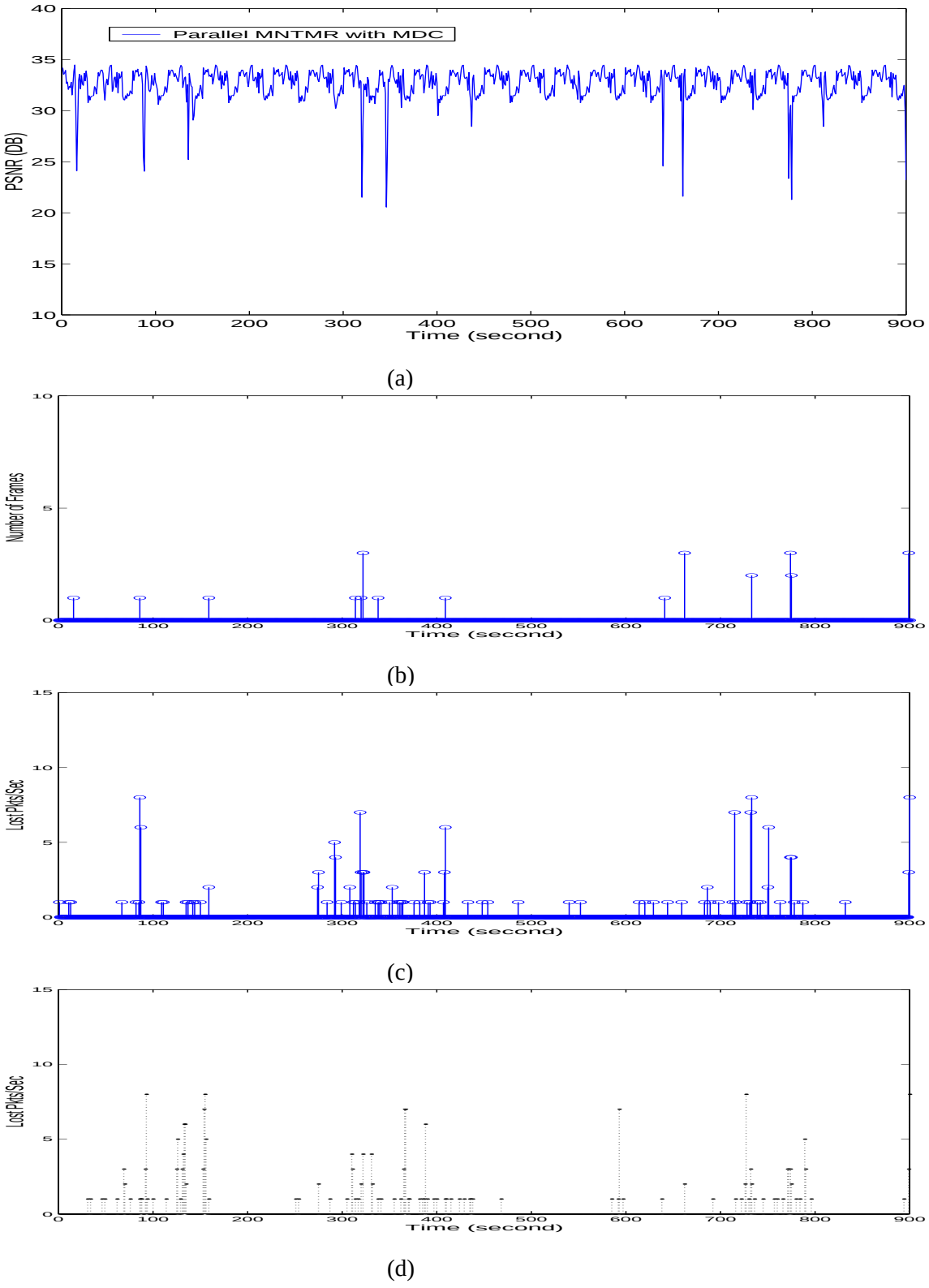
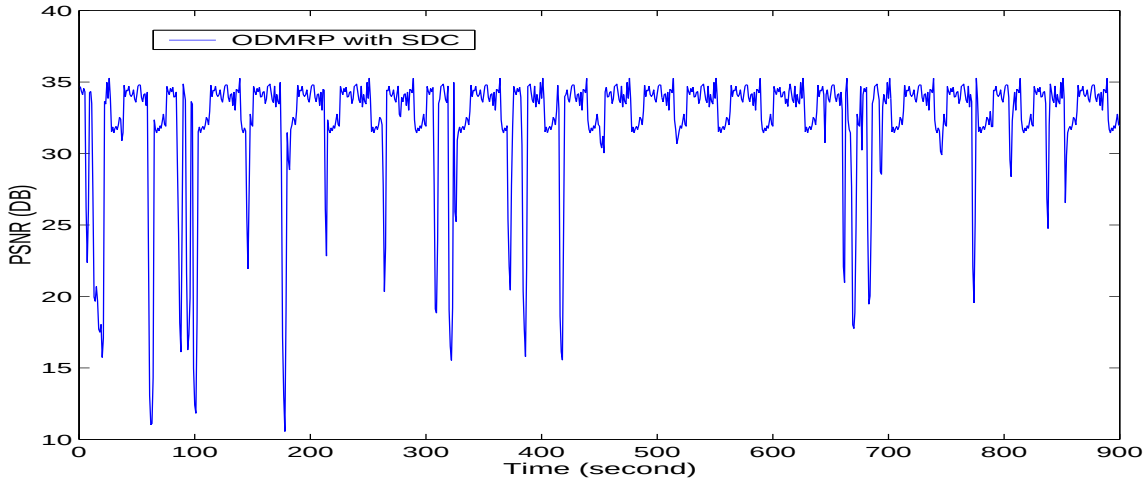
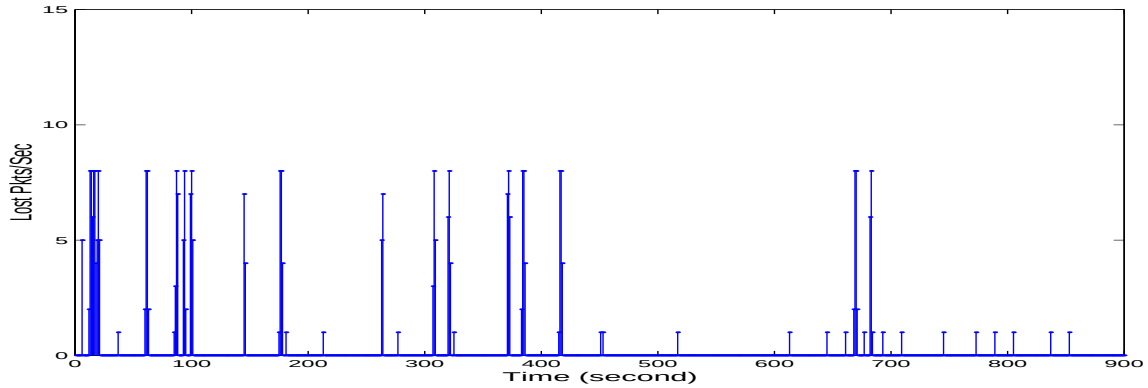


Fig. 8. (a) PSNR of the received frames using Parallel MNTMR and MDC; (b) Number of Frames that both descriptions are lost; (c) Lost packets per second for substream 0; (d) Lost packets per second for substream 1.

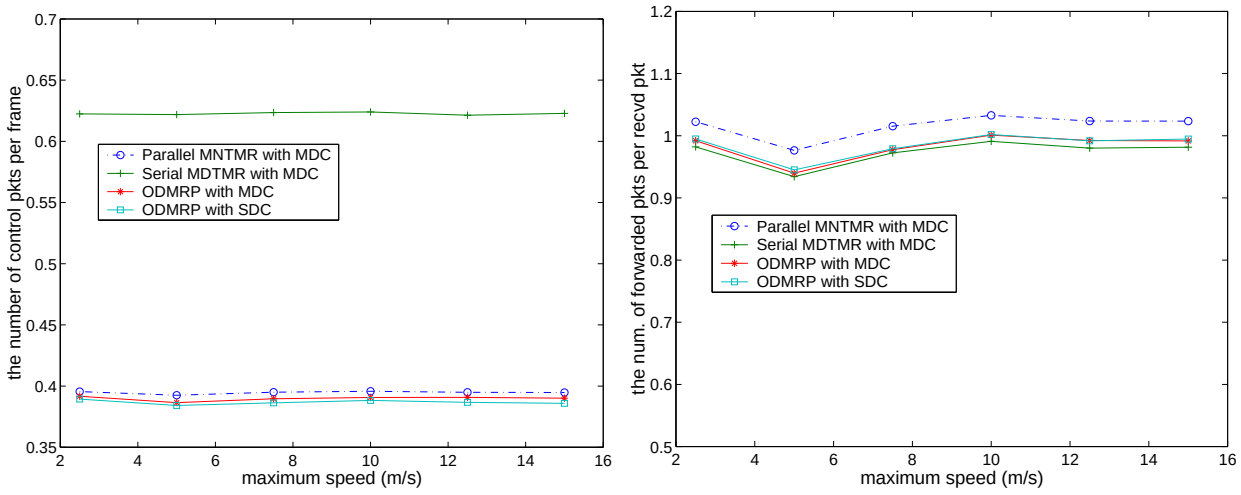


(a)



(b)

Fig. 9. (a) PSNR of the received frames using ODMRP and SDC; (b) Lost packets per second for the stream.



(a) (b)

Fig. 10. Performance evaluation for multiple tree protocols: (a) The normalized control packets; (b) The normalized forwarded data packets.

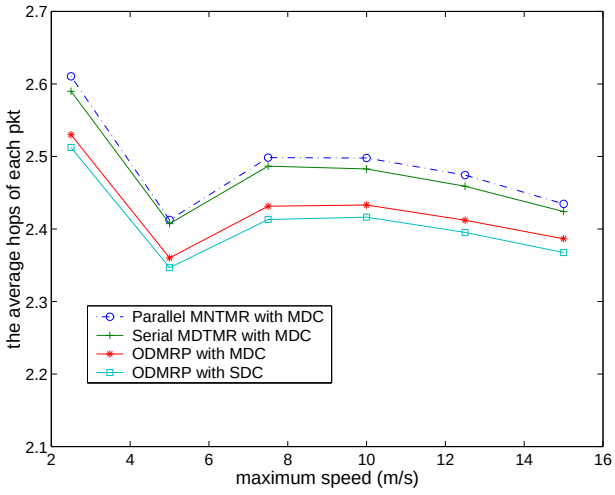
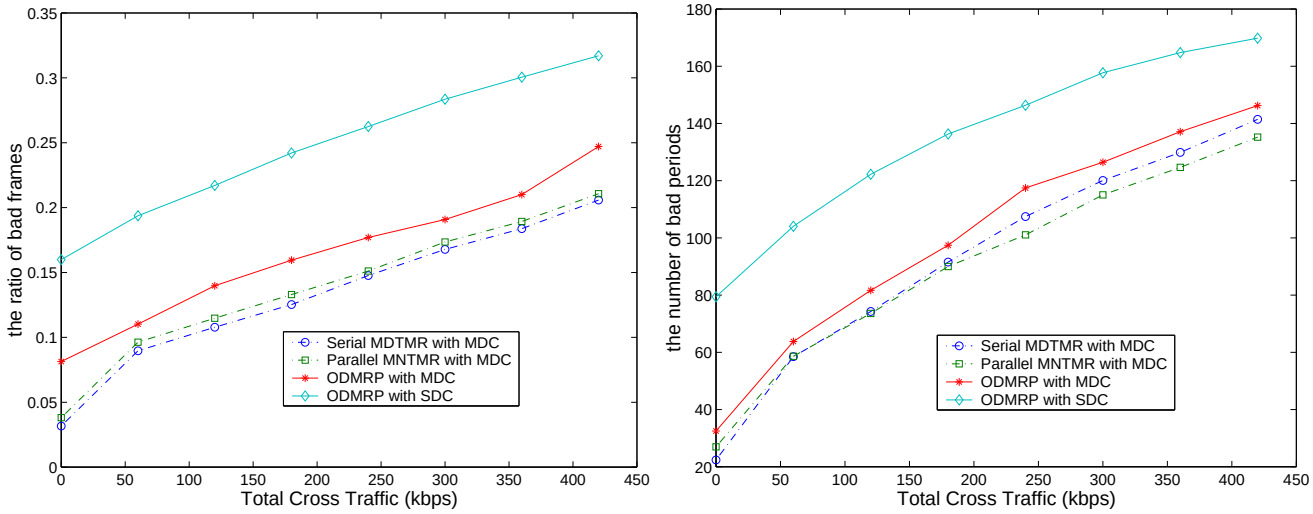
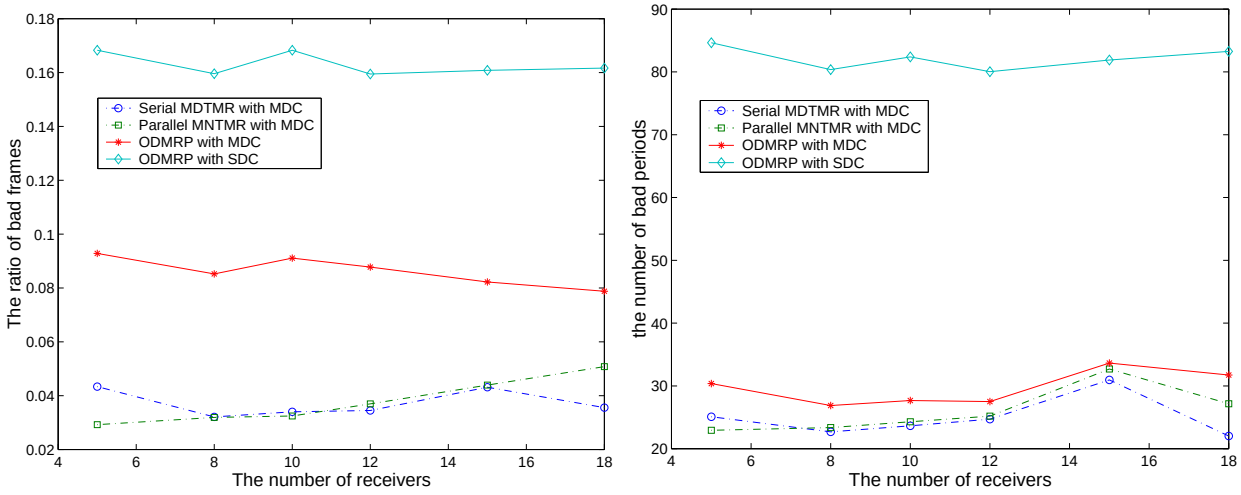


Fig. 11. Performance evaluation for multiple tree protocols: The averaged number of hops.



(a) (b)

Fig. 12. Performance evaluation for multiple tree video multicast with cross traffic: (a) The ratio of bad frames; (b)The number of bad periods.



(a) (b)

Fig. 13. Performance evaluation for multiple tree video multicast with varying number of receivers: (a) The ratio of bad frames; (b)The number of bad periods.

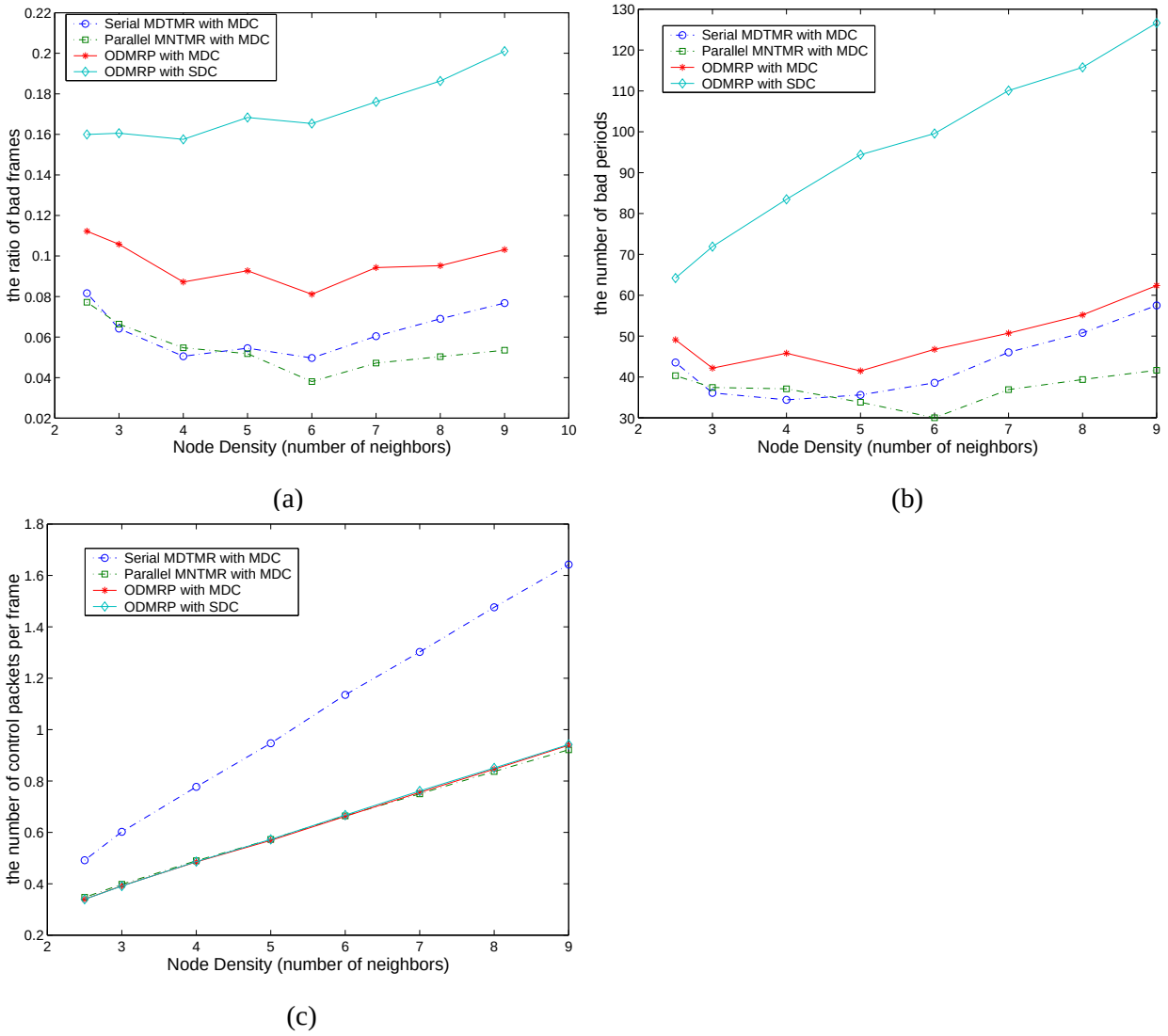


Fig. 14. Performance evaluation for multiple tree video multicast with varying node density: (a) The ratio of bad frames; (b)The number of bad periods; (c) The normalized control packets.

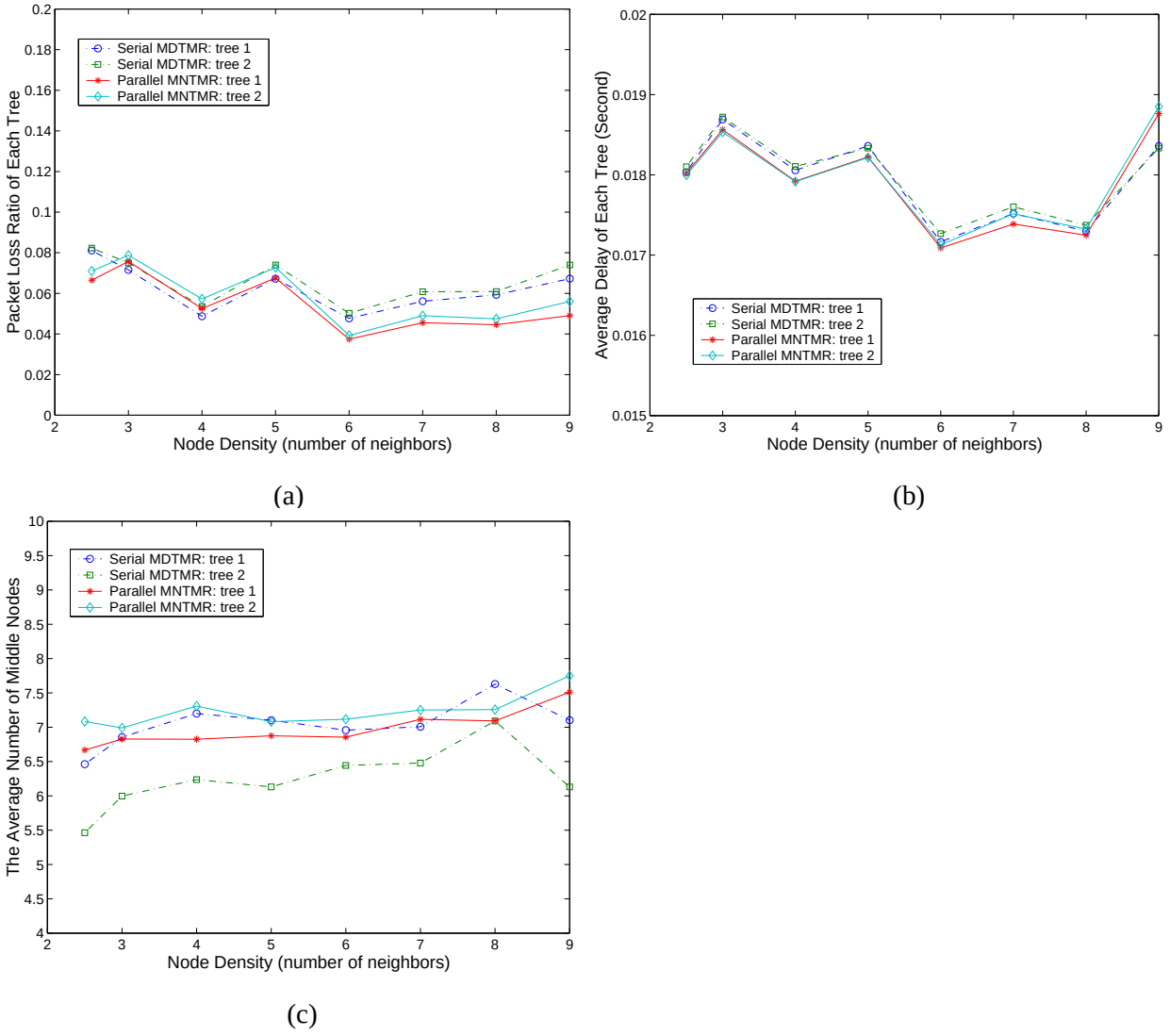


Fig. 15. Performance evaluation for multiple tree video multicast with varying node density: (a) Packet Loss Rate of Each Tree; (b)Average Delay of Each Tree; (c) Average Number of Middle Nodes of Each Tree.