Team Oriented Multicast: a Scalable Routing Protocol for Large Mobile Networks[*]

Yunjung Yi, Mario Gerla and Joon-Sang Park
Computer Science
University of California at Los Angeles
California, USA
{yjyi, gerla, jspark}@cs.ucla.edu

ABSTRACT

This paper proposes a multicast protocol, called *Team Oriented Multicast* (TOM). TOM builds up a "motion aware hierarchy to support efficient, scalable team multicast protocol. TOM identifies clusters of nodes with same affinity as teams and manages the multicast membership information using the unit of team rather than dealing with individual node members. TOM uses a two-tier data dissemination approach where the source propagates a data packet to each subscribed teams leader and each leader forwards the data to the entire team. TOM constructs a multicast mesh structure among leaders of subscribed teams, where each leader is connected to m other parent leaders, receiving duplicate packet streams from each parent. Each team leader proactively maintains the list of nodes in the same multicast mesh.

The main contributions of TOM (which set it apart from general hierarchical multicasting protocols) are (1) TOM provides an efficient and scalable multicasting protocol exploiting affinity team model, which is a realistic model for many MANET applications and makes the mobility management much simpler; (2) using a m-ary mesh structure and multi-path neighbor aggregation technique, TOM provides a highly reliable data transmission platform with fairly low redundancy overhead; (3) the proactive maintenance of membership information between team leaders can supply useful QoS information to multimedia applications.

Simulation results show the effectiveness, scalability and reliability of TOM in various representative scenarios.

1. INTRODUCTION

With the advances in wireless ad hoc communications, robotics and microflyer technology, the deployment of large-scale networks with hundreds and even thousands of distributed autonomous nodes will be possible in the near future. In such large scale networks, with no fixed infrastructure, providing an efficient, scalable routing and multicast scheme is extremely challenging. Indeed, many potential applications in mobile ad hoc networks require multicasting. For example, collaborative, distributed mobile computing applications (e.g., sensor networks, workgroups), disaster relief operations and war front activities all heavily depend on multicasting to exchange data among multiple participants. In most of the above multicast applications, QoS support such as real-time data transmission and reliable packet delivery are essential. In addition, because of wireless bandwidth limitations, the underlying multicasting protocol itself must be very efficient, i.e., work with low line overhead, provide highly reliable data transmission, and be scalable. A unique requirement of multicast QoS (which makes it more difficult than Unicast QoS) is to effectively scale to a large number of receivers. In [17], the authors have shown that a hierarchical routing is essential to achieve adequate performance in very large networks. A hierarchical approach, where multicast group receivers are grouped into a few clusters, can be exploited if a stable cluster platform can be maintained. By grouping receivers, QoS protocols consider only a small number of representative nodes instead of thousands of individual members. However, the assumption of a stable cluster platform often fails in MANET scenarios where nodes move quickly and thus the membership of a cluster is extremely fragile. With an unstable cluster structure, hierarchical multicasting may not be a good solution due to excessive cluster maintenance cost.

That observation leads us to conclude that developing a hierarchical multicasting protocol working for all possible scenarios is probably not feasible. Fortunately, in many large scale MANET scenarios (e.g., warfront activities, search and rescue, disaster relief operations, etc.), the mobile nodes are organized in teams with different tasks and, correspondingly, different functional and operational characteristics. In particular, nodes in the same team will have in coordinated motion. We call this model the "affinity team model". For example, attendees of a major conference can be subdivided into teams based on their topic interests for the purpose of organizing birds of a feather sessions; various units in a division can be organized into companies and then further partitioned into task forces based on their assignments in the battlefield. In a highway, platoons of cars can be

[*]UCLA TR 03003

treated as a team because of their motion affinity. Other examples are search and rescue operations, disaster monitoring, and mobile sensor platforms. Our basic observation of those applications is that nodes can be grouped based on their physical location, mobility, or interests. With the affinity team model, it suffices for mobility management to keep track of only one of the nodes in each team (a representative node). Other nodes in the team can be reached through the representative node. As our affinity team model guarantees the stability of clustering (teams) in some degree, the design of an efficient scalable hierarchical multicast structure is now realistic.

Our proposed idea, *Team-Oriented Multicast* (TOM), exploits the affinity team model. It defines teams and manages the membership information using the unit of team rather than that of a set of individual nodes. A team is defined as a set of nodes that have the motion affinity and interests differentiated by subscribed multicast groups. To fully utilize that logical hierarchy of teams, TOM provides a two-tier multicasting approach where the source propagates a data packet to each subscribed team‘s leader and each leader forwards the data to the entire team. As one can easily expect, the performance of such a two-tier approach considerably depends on the design of first-tier communication platform among leaders. From now on, we will call the leader the team representative node (TRN). If the reliability and latency of data transmission to each TRN can be bounded, this two-tier approach can provide a reasonable throughput. Otherwise, this approach may perform worse than a flat multicast protocol such as ODMRP [18], because of the extra overhead to manage the logical cluster architecture. In Internet multicast, shared tree structures are often used to improve the efficiency of multicasting. Internet multicasting protocols emphasize efficiency rather than reliability because the underlying wired medium guarantees the data delivery in some degree. In MANET scenarios, this is not true anymore. Due to collisions, congestion, link errors, jamming, asynchronous links and interferences, the delivery ratio on a wireless connection varies over time and it may becomes unacceptable(e.g., less than 60%) [9]. The delivery ratio of a packet sharply drops as the traveled hops increase [9]. Since our target systems are pure “flat networks, and thus, the average hop distance between two nodes increases as the network size grows, a hierarchical scalable routing protocol should cope with such a low success rate of data reception. This unique characteristic makes the hierarchical MANET multicasting protocol distinctive from hierarchical multicasting protocols proposed in wired network. Thus, the main focus of TOM is to provide an efficient and robust platform among selected team leaders.

The expected contributions of TOM are as follows:

- a scalable multicasting protocol: TOM is scalable as the network size, group size and group numbers increase. TOM realizes the scalability via hierarchical structure and aggregated group maintenance.
- a redundant, reliable data dissemination structure: TOM provides a m-ary mesh structure between team leaders where each leader has at most m connections to other leaders, receiving replicated packet streams from them.

  As a difference from other multicasting protocols based on a mesh structure [18, 23], the cost of TOM multicast mesh construction is very low. Furthermore, TOM restricts the number of retransmissions to m-1 for efficiency. Unlike other schemes, where a flooding mechanism is generally used to build and maintain a mesh structure, TOM does not incur control packet flooding to find a mesh structure. Rather, in TOM, each node proactively maintains the partial information of multicast groups so that a new team‘s leader can locally find a contact node to send a Join Query.
- a reliable and efficient data transmission mechanism: TOM proposes a multi-path neighbor aggregation technique, which leads to potential r multiple paths between two nodes in the multicast mesh. With the multi-path scheme and the virtual forwarding structure, TOM efficiently and reliably forwards a data packet through the multicast mesh structure.
- multiple sender/multiple receivers: TOM assumes dynamic multiple senders in a multicast group. Any node in a group can be a source node. Even a node not belonging to a group can be a source.
- fault tolerant: TOM targets a fault-tolerant protocol protected against the failure of any node, especially team leaders.

The rest of paper is organized as follows. Section 2 briefly overviews the related works. In Section 3, we will discuss the design issue of TOM, and the detailed protocol description will follow in Section 4. Following Section 5 will show the evaluation of TOM through simulation study. Finally, we conclude our paper in Section 6.

2. RELATED WORKS

As the node mobility is one of main challenges to design MANET routing protocol, many researches have been conducted to develop a mobility model [5, 8]. The observation of group affinity is not new. In [5, 14, 21], the author already proposed a group mobility model where a set of nodes move together. There are many researches on clustering algorithms and routing algorithms considering node mobility [6, 2, 12, 10]. However, not many researches have been accomplished on hierarchical MANET multicasting protocols working with group mobility.

In the wired network, a book of hierarchical multicasting protocols have been proposed for the purpose of scalability [3, 22]. As mentioned earlier, the hierarchical multicast protocols in wired network favors the efficiency of multicasting. As our targeting scenarios have different characteristics, those protocols can not be directly applicable to MANET environment.

A few MANET multicasting protocols choose hierarchical approaches [16, 23, 7, 26]. CAMP [16] extends the basic approach of the core-based tree (CBT) [4] protocol for the creation of multicast structure with an allowance of one or multiple cores. “Cores” can limit control overhead for members to join the multicast groups. With deploying one or multiple Cores for each multicast mesh, CAMP can reduce flooding overhead for Join Request packets. However, CAMP does not provide a robust and efficient data forwarding platform between core nodes. Rather, CAMP depends on the underlying proactive unicast protocol (bellmand-ford routing scheme) to propagate a data packet to cores and members, which limits the scalability of CAMP due to the huge memory requirement and heavy routing overhead [13].

AMRoute uses underlying unicast routing protocol to build up a multicast tree. Each group has at least one logical core that maintains the multicast structure. Cores periodically send Join Requests and members send Join Reply. In AMRoute, however, the actual multicasting follows a multicast mesh structure rather than through logical cores i.e., the logical cores are not central nodes to forward a packet. Thus, TOM is divergent from AMRoute.

MCEDAR [23] constructs the set of *cores*, which is a set of dominant nodes that covers all the members in a multicast group. By constructing a multicast mesh among only *cores* and allowing cores to re-broadcast the data packet, where a core covers a set of members, MCEDAR reduces membership maintenance cost and forwarding overhead.

Those ideas have been mostly focused on the efficiency and reliability in a rather small scale network. Unicast tunneling used in AMRoute and unicast transmission in MCEDAR are not scalable, since the cost of unicast grows as the number of participants or *cores* increase.

In [26], the authors proposed a hierarchical multicasting based on the scalable unicast routing LANMAR [10], called M-LANMAR. The approach and design goals of M-LANMAR are similar to TOM in that M-LANMAR proposed a two-tier data propagation exploiting group mobility. TOM, however, is divergent from M-LANMAR in that TOM provides a reliable, robust and efficient first-tier communication platform, whereas M-LANMAR totally depends on the underlying unicast protocol to propagate the packet to landmarks, and thus it shows the limited scalability.

TOM has different objectives from previous multicast protocols. The main objective of TOM is to provide a multicasting paradigm supporting a potential QoS extension in a large mobile network. As one of the main challenges of a scalable multicasting is to handle a huge number of receivers, TOM considerably removes the burden of scalability by building a virtual hierarchy and allowing only leaders visible from outside. Further, TOM addresses the low packet reception rate in a large network and provides a robust forwarding structure. This is important especially in a large-scale network where the cost (e.g., latency and packet overhead) of packet recovery is considerably high.

3. DESIGN RATIONALE

As a first step to a hierarchical multicasting, TOM constructs a virtual hierarchy by organizing nodes to a few teams based on affinity team model and selecting a leader for each team. With such a hierarchy, TOM provides a two-tier multicasting paradigm where the source delivers the packet to each member in two steps: (1) inter-team data forwarding: data forwarding to each team leader called a team representative node (TRN) and (2) intra-team forwarding: data dissemination within a team initiated by the TRN node.

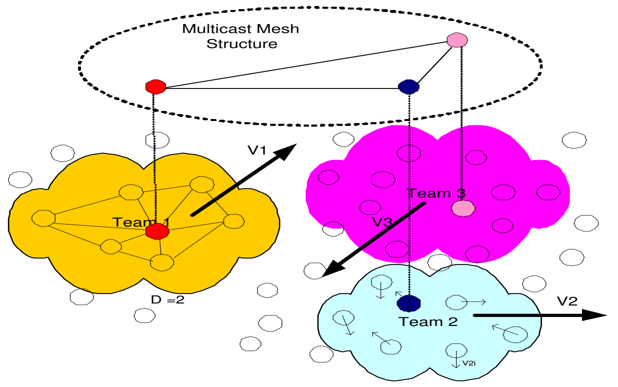

Figure 1: Target System Overview of TOM protocol

The main components of TOM are as follows: (1) team definition and discovery; (2) *inter-team group membership management*: TOM builds a multicast mesh among subscribed leaders; (3) *intra-team group membership management*; (4) *inter-team data forwarding*: TOM builds a virtual forwarding structure to propagate a data through the multicast mesh structure; (5) *intra-team data forwarding*.

The overview of our target system is shown in Fig. 1. The network consists of several teams $\{T\}$ and individual nodes that do not belong to any team due to the lack of affinity. A team T is a connected un-directed graph with the maximum distance D from a node i to j (i and $j \in T$). A link (i, j) implies a direct connection between i and j. A team T is defined as a set of nodes satisfying following conditions:

- same mobility pattern: Our affinity team model assumes that the mobility of a team can be specified as a vector with velocity and direction (see Fig. 1, each team has a mobility Vi). Each node in a team can randomly move within a bounded area ($d * TX$, where $d = \frac{D}{2}$ and TX is the transmission range). Thus, a set of one-hop neighbor nodes of each node may be dynamic, but a set of D-hop neighbor nodes of each node in the same team is stable.

- common multicast group membership: Any two nodes in the same team should have subscribed at least one common multicast group. By subscribing a common group, nodes in the same team share the interest.

- D-hop reachability: Any two nodes in the team are mutually reachable within D-hop distance.

Each node discovers a team and selects a leader in a distributed manner similar to a clustering mechanism [6, 12]. The detail description of full team discovery protocol is out of scope of the paper. Current team management algorithm is based on the idea proposed in [12].

In the paper, for the sake of simplicity, we assume followings:

- a node does not join a multicast group if it does not belong to a team
- all nodes in the same team subscribe the same multicast groups

Since we assume the atomic property of a team (i.e., all nodes subscribe the same team), inter-team membership maintenance and data forwarding become simple. Thus, this paper focuses on inter-team membership management and data forwarding.

3.1 Inter-team Group Membership Management

Many multicast routing protocols [18, 20] are source-driven approaches in that they build up a source tree rooted at the source node. Our interests also cover some applications where source information is not available prior or the number of sources is comparably large so that the source-tree approach is not useful. For example, in a publisher-subscriber system, often publishers are not deterministic, rather dynamic. More importantly, the main goal of TOM is to provide a highly scalable and reliable multicasting protocol. With those requirements, TOM builds up a m-ary connected multicast mesh structure among subscribed teams‘ leaders. In m-ary connected multicast mesh structure, each leader has at most m undirected connections with other leaders and any two leaders in the structure are mutually reachable. By allowing m redundant packet receptions from connected leaders, note that each node forwards a data packet to all connected leaders except toward incoming direction, our mesh structure provides a reliable transmission platform over a tree structure. The parameter m is used to provide the load balancing among leaders. A leader is allowed to connect to at most m other nodes, and thus, the burden of forwarding at each leader can be bounded. To effectively manage the mesh structure with dynamic membership changes, TOM develops a mesh maintenance algorithm, where the goals of algorithm are (1) requiring less dynamic mesh re-construction; (2) working in a distributed fashion; and (3) demanding low overhead.

To maintain a path between two leaders connected in the multicast mesh structure, TOM uses a distance vector routing protocol (DSDV). The reasons why we choose the proactive route maintenance are as follows: (1) as we use a broadcast mechanism to deliver a packet, a node cannot detect a link failure usually perceived through a unicast packet delivery failure at MAC layer. Thus a proactive maintenance is essential; (2) our targeting scenarios are very dynamic so routes are fragile. The recovery overhead and latency of on-demand routing may be not tolerable, especially in our large scale system with many on going sessions; (3) the number of leaders is comparably small so that the overhead of proactive maintenance is manageable.

With random mobility of each team, all nodes should proactively manage the paths to leaders. Thus, each node in the network maintains the table of all the leaders who subscribed to any group and periodically exchanges and updates that table with neighbor nodes. We call the table of leaders as TRN table hereafter.

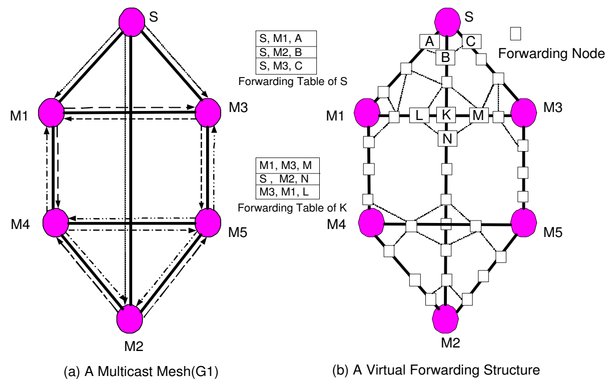

Figure 2: Multicast Mesh and Virtual Forwarding Structure

3.2 Inter-team Data Forwarding

Inter-team data forwarding mechanism is the key to the success of TOM‘s two-tier data transmission approach. To design an efficient and reliable inter-team forwarding mechanism, however, is very challenging, since the average distance of data transmission is very large and the path reliability is extremely low. The path redundancy by the mesh structure does not significantly improve the reliability without a bounded packet reception rate between two connected leaders. Thus, our main goal of inter-team data forwarding scheme is to improve the path reliability in an efficient way.

The easiest and simplest way is to use separate unicast tunneling between two connected leaders to transfer a data. Since unicast is more reliable than broadcast with IEEE 802.11 DCF protocol where unicast packet uses RTS/CTS handshaking, acknowledgement and retransmission upon a failure, this approach provides comparably reliable data delivery to each leader. However, unicast tunneling for the multicast data transmission cannot exploit the advantage of shared broadcast medium where a broadcast transmission covers the set of neighbor nodes and potential duplicate transmissions from different neighbors improve the packet reception rate at a node. More seriously, unicast tunneling has following potential drawbacks: it results in (1) fluctuating and generally large packet latency compared to the broadcast mechanism: since a unicast transmission requires three-way handshaking with the next hop, the latency increases as the offered load at the next hop node as well as at this node increases. We will show the latency of unicast tunneling through the simulation study; (2) starving: a node may starve if the next hop is fully saturated; (3) ineffectiveness: a packet to the saturated next hop i.e., starving packet prevents transmissions of other packets to different neighbors in the same node.

With the broadcast mechanism, the mentioned problems can be lessened. We, thus, choose the broadcast mechanism to forward a multicast data packet. However, the reliability of a broadcasted packet is considerably lower than that of a unicasted packet. Even worse, the reliability becomes lower as the traveling distance grows. To achieve a high reliable data transmission using the broadcast mechanism, TOM proposes a multi-path scheme.

More precisely, TOM proposes the multi-path neighbor aggregation (MPNA) technique. MPNA builds a virtual forwarding structure including intermediate nodes and leaders in the multicast mesh. Fig. 2 illustrates an example of a multicast mesh structure and following a virtual forwarding structure. Ideally, a node in the virtual forwarding structure should relay a packet only once for efficiency. In MPNA scheme, however, a node may relay the packet more than once to propagate aggregation information, if necessary. The detailed description will follow in next section. We should note that this forwarding node concept is not new. It was already proposed in ODMRP. However, MPNA develops a different mechanism to find forwarding nodes. In ODMRP, each intermediate node sets the forwarding flag, if it receives a Join Reply packet from a neighbor, thus explicit control messages are necessary. In MPNA a sender calculates next forwarding nodes (next hops) using TRN table and adds the aggregation header piggybacking the information to the packet. A node sets the forwarding flag, if it discovers that, by examining the aggregation header of incoming packet, a previous hop selects it as a next hop. Conceptually, it is more similar to soft-state Differential Destination Multicast (DDM) [15] than to ODMRP. In DDM, targeting a small group, each source aggregates the packet in a similar way to MPNA. DDM, however, attempts to reduce the aggregation information by deploying the synchronization between a node and the next hop. If a route is pretty stable, DDM can significantly reduce the aggregation overhead. TOM, however, is designed for a network with high mobility, and thus, the stability of a path is pretty low. More importantly, the underlying routing protocol used to update paths between leaders, DSDV, tends to change routes frequently. With DSDV, a node updates a path whenever it discovers a fresher route even though the current path is still valid [9]. Thus, the optimization of DDM is not directly applicable to TOM. Furthermore, TOM provides multi-path transmission. Thus, it differs from previous schemes [18, 15].

4. PROTOCOL DESCRIPTION

In this section, we present the detailed algorithms for mesh maintenance and multi-path neighbor aggregation scheme.

4.1 Multicast Mesh Maintenance

Our mesh structure is an undirected connected graph $G = (V, E)$. Each vertex $v \in V$ can have at most m edges. The redundancy factor m can be adjusted considering the overall reliability. However, to satisfy the connectivity of our graph model, m should be greater than 2 (two) (see Appendix.A2). A v forwards an incoming packet from v_s to all other connected nodes v_d such that $\exists\ e = (v, v_d) \in E$ and $v \neq v_s$ (see Fig. 2 (a)). As default, we use $m = 3$. Each vertex v in the graph has a unique sequence number seq_v, which is assigned at Membership Join phase. A root vertex $r \in V$, which has the lowest sequence number among V, maintains the vertices list V and the current sequence number $C(seq)_G$ to assign a new member. The sequence number is important to maintain a connected graph with dynamic membership changes i.e., new join, leave or link changes.

4.1.1 *A Group Membership Join*

Many existing multicast protocols (e.g., MAODV [20], MCEDAR [23]) use flooding mechanism to join a multicast group, since a new member does not have the information of a multicast structure. Distinctively, TOM, because of TRN table update mechanism, can propagate the partial membership information with low overhead to the entire network. Only root vertices of multicast groups advertise the group address and the size of the multicast mesh graph to the entire network by piggybacking the information on TRN table exchange messages so that a new team can finds a point to send a Join Query by looking up its TRN table.

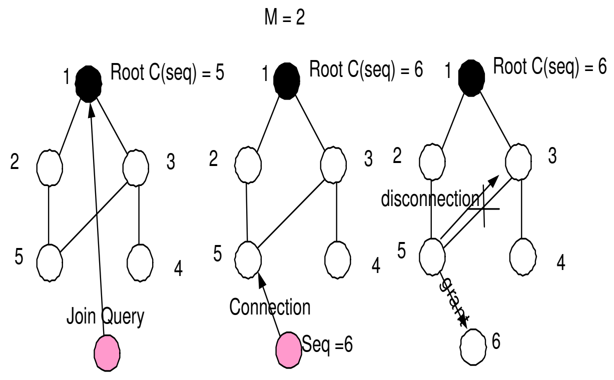

Figure 3: Join Procedure

The Join of a new team T_i to a group m_j is a procedure to add a vertex trn_i (the leader of T_i) and edges with the minimal cost to the multicast mesh graph $G(m_j)$ while keeping $G(m_j)$ connected. When a new team T_i wants to join a multicast group m_j, the leader trn_i of T_i first looks up its local TRN table to retrieve the root vertex of $G(m_j)$ and sends a query if available. Otherwise, (i.e., this node is a new incoming node or no team has subscribed to m_j), trn_i claims itself as a root vertex in a graph $G = (\{trn_i\}, \emptyset)$ and starts advertising the membership information with TRN table exchange. Once a root vertex discovers another graph for the same group, it tries to merge two graphs (**Graph Merge Procedure**).

When a root vertex r receives the query packet, it increments the current sequence number $C(seq)_G$ and assign to trn_i (i.e., $seq_{trn_i} = C(seq)_G$). r returns the member list V and new sequence number seq_{trn_i} to trn_i. Each node has two connection list: the parents list CL_p and children list CL_c. For each link $(v,\ w)$ where $seq_v < seq_w$, v is a parent of w and w is a child of v. To guarantee a connected graph, a vertex v should have at least one link $e_p = (v,\ w)$ where $seq_w < seq_v$ (i.e., $CL_p \neq \emptyset$) (The proof is given in Appendix A.1). trn_i sorts the member list V in ascending order according to the distance from trn_i based on its TRN table. Until, trn_i finds a parent node to connect, it sends a Connection Request to a node v_j i.e., j-th element in V. Upon receiving a Connection Request packet, v_j performs **Connection Establish Procedure** as follows. Without a link and node failure, trn_i will find at least one parent node if $m \geq 2$ (see Appendix A.2). Note that we assume that network is not partitioned.

Once trn_i is connected to G, then trn_i informs the root vertex r. The root vertex adds trn_i to V and propagates to G with the current sequence number $C(seq)_G$. To provide resilient membership maintenance in spite of a failure of the root, we duplicate the membership information to each vertex in the graph. Fig. 3 illustrates an example of Join Procedure. Once a node joins a group, it may add connections up to m links adaptively.

- Connection Establish Procedure (three-way handshaking): Upon the Connection Request from trn_i, v_j reacts differently based on its degree. If its degree is less than m, v_j simply grants the Connection Request. trn_i and v_j uses three-way handshaking to confirm the connection. If v_j has m connections and $|CL_p| > 1$, v_j removes the connection to the furthest vertex v_p by making Disconnection Request. v_p performs **Disconnection Procedure** and replies back to v_j. With a reply from v_p, v_j grants the connection.

  If v_j has only one parent, v_j sends **Connection Reject** reply to prevent a potential disconnection of the mesh structure. As shown in Appendix. A1, a vertex should have a connection to another parent vertex to preserve the connectivity in spite of dynamic removal/add of edges.

  If the connection between trn_i and v_j has established, both trn_i and v_j propagate updated their CL_p and CL_c to the entire team respectively to maintain a consistent connection list to each node in the same team for the purpose of fault tolerance.

  To recover from a lost packet, trn_i uses a several retransmission retrials.

- Disconnection Procedure: When w receives Disconnection Request packet, w accordingly removes v from $CL_p(w)$ or $CL_c(w)$ and sends a reply back to v. And, w recovers its connectivity if $CL_p(w) = \emptyset$ by finding and connecting to another parent node. If w fails to find such y, it removes all edges and initiates Join Procedure again.

- Connection Maintenance Procedure: With periodic TRN table exchange, each leader monitors the liveliness of connected leaders. Without update from w for a timeout T, v removes w from the connection list. If CL_p $= \emptyset$, v finds another parent node to connect. If the root fails, the node with the lowest sequence number among live nodes becomes the new root. The new root r informs the change of a root vertex to all leaders in the multicast mesh. Each leader removes w from the connection lists if $seq_w < seq_r$.

- Graph Merge Procedure: The advertisement of a root node allows another root to discover multiple graphs for one multicast group. When a root vertex r_1 of a graph G_1 discovers another graph G_2, r_1 initiates Graph Merge Process by sending a Group Merge Request to r_2. If G_2 is merged to anther graph already, r_2 relays the request to its root node. If r_2 leaves the group, r_2 replies Rejection. A graph with fewer vertices G_1 is merged into the other G_2. Tie breaks with node ID. The root of merged graph r_1 performs Membership Join Procedure to another graph (i.e., it gets the new sequence number from r_2). All nodes in the merged graph updates the sequence number seq_v $= seq_v + (seq^{new}_{r1}$ - $seq^{old}_{r1})$ and the root ID. For the potential graph merge, a root allows only m -1 edges.

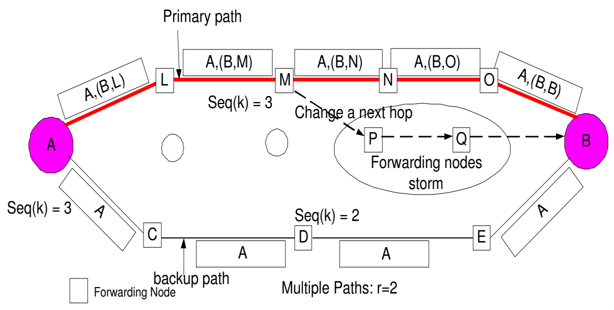

Figure 4: Multi-path Example

4.1.2 *Membership Leave*

When a team leaves a group, the leader sends an explicit Membership Leave Request. If the leader is not a root vertex, it disconnects all connections and informs the root vertex. If a root vertex wants to leave, it chooses the vertex v with the smallest sequence number and hands over the root role. A new root advertises the entire nodes in the graph the change of root address. A root without edge simply stops advertising so that each node in the network removes the entry from TRN table after a timeout.

4.1.3 *A Special Case: 1-level multicast tree*

With $m = 0$ i.e., no edge in the graph, a 1-level multicast tree is implicitly built for every source. In this case each team‘s leader claims the root role without graph merge and it thus advertises its membership to the whole network. In 1-level multicast tree, a source will separately transmit the data to each subscribed leader. This 1-level multicast tree has pros and cons. Because of no relationship between subscribed teams, the failure of teams does not affect other teams. And, the multicast join and leave operations are much easier than m-ary mesh structure because of no mesh maintenance cost. However, this 1-level multicast tree has a limiting factor of the scalability as the number of subscribed teams grows. Since the traffic is concentrated on nodes near to the source node, the efficiency sharply degrades as the number of receivers increases. Moreover, a single connection between the source to each leader will lead a poor delivery ratio as the average distance increases. In our simulation study, we will compare the throughputs of m-ary multicast mesh and 1-level multicast tree structures.

4.2 Multi-Path Neighbor Aggregation

In this section, we present multi-path neighbor aggregation technique used to deliver a data packet between connected leaders in the multicast mesh. With the aid of root vertices, an outsider node i.e., not a member node can multicasts a data to a group. In that case, the source sends a packet to the root vertex and root vertex initiates the inter-team data forwarding. Otherwise, a source node transmits the data packet to its leader. And the leader initiates the forwarding to other leaders.

The main purpose of MPNA is to provide an effective mechanism to build a forwarding structure without incurring extra overhead as mentioned earlier. So, MPNA exploits outgoing data packets to assign necessary forwarding nodes. The basic idea is that a previous hop node assigns the next hop addresses required to deliver the packet to designated destinations by creating aggregation entries.

In MPNA, each node creates an aggregated packet for possibly multiple destinations before transmitting a packet and broadcasts to its neighbor nodes. To inform designated neighbors that should receive and relay the packet, it creates the aggregation header which includes the set of the destination and next hop. For example, the packet from S in Fig. 2 includes the aggregation information (M1,A), (M2,B) and (M3,C). The information included in the aggregation header for a packet is as follows: the source address s_{m_j}, the group address m_j, the sender address s, the number of AGG_ENTRYs numAGG, and the set of aggregation entries $\{AGG\_ENTRY\}$. Only leaders can be senders. A leader in the mesh structure s initiates the aggregation for destinations d_k in the mesh structure. A AGG_ENTRY consists of: (s, d_k, next hop address $next\_hop_k$, a control sequence number for $seq(k)$).

Each node n in the virtual forwarding structure has a forwarding table F. n has an entry for each connection that goes through n. For example, node K will have three entries for (S, M2), (M1, M3) and (M3, M1) in Fig. 2. Note that, even though the multicast mesh is an undirected graph, the data transmission has a direction (e.g., M2 $\rightarrow$ M1 and M1 $\rightarrow$ M2). Thus, each node separately manages entries for the links (s,d) and (d,s). Now, let a connection $k = (s_k \rightarrow d_k)$ of a multicast group m_j. An entry for k, say f_k, in the forwarding table of n includes: ($s = s_k$, $d = d_k$, m_j, seq, next hop address p_n, forwarding flag $flag$, aggregation flag agg, update time t, forwarding force flag $force$). In Fig. 2, we illustrate a brief version of forwarding table of K, where a solid line presents the primary path. Node K has three forwarding entries for three connections with the next hop L, M and N respectively.

Each field in the entry f_k of forwarding table F is needed for following reasons:

- sequence number seq: The sequence number for each connection k, $seq(k)$ is used to determine a primary path. As MPNA allows multiple paths, a scheme is required to differentiate the primary path and backup routes. Using the sequence number incremented on the sender s_k, an intermediate node n assumes that this node is in the primary path, if $seq(k)$ of the sender is equal to $f_k.seq$ (a sequence number in the forwarding table for a connection k) and $next\_hop_k$ is n (see Fig. 4). Only forwarding nodes in the primary path aggregate the information of the next hop and the destination d_k for a connection k. In Fig. 4, the packet through the primary path L-M-N-O includes the aggregation information regarding the destination B, but the packet through a backup path C-D-E does not include any aggregation information. Note that, if a node aggregates a next hop h for a destination d, h will set its forwarding flag. Thus, by allowing only nodes in the primary path to assign new forwarding nodes, we limit the number of forwarding nodes. Further, we limits the next hop change at each intermediate node to prevent a potential forwarding nodes storm effect where an intermediate node in the primary path makes a branch (by changing the next hop), so thus there are more than two primary paths are existing (see Fig. 4, if a node N changes the next hop from M to P, nodes M and P will be forwarding nodes and assign next hops O and Q respectively). Each intermediate node can change the next hop for the destination d_k, only when the sender s_k increments the sequence number. A sender s_k increments the sequence number $seq(k)$ at every interval I_{update} (< TRN table update period). With a new sequence number, s_k and each intermediate node k_i updates the next hop address.

- aggregation flag agg: As long as agg is TRUE, this node is in the primary path between s_k and d_k, and thus this node aggregates the information regarding d_k when it relays a packet.

- forwarding flag $flag$: A forwarding node for a connection k sets the flag. A node, once setting the forwarding flag $flag$ i.e., becoming a forwarding node for a connection k, forwards a packet until it unset $flag$. To allow r multiple paths, in MPNA, a forwarding node unset $flag$, only if (1) it is not selected as a next hop for more than r-1 rounds by monitoring sequence number $seq(k)$ in the incoming packet from s_k; (2) the forwarding entry is not updated for T_{update}.

- forwarding force flag $force$: A node sets this flag when the forwarding entry is newly created, i.e., its selected as a new next hop for d_k. Thus, this node should relays the incoming packet to finish the forwarding node selection to reach d_k. For example, node K in Fig. 2 receives a duplicate packet from M1 after it forwards the packet from S to M2, K is required to forward the packet to inform M to be a forwarding node.

Incoming packet processing

To create or remove the entries from the forwarding table, each intermediate node processes every incoming packet. A node n process the aggregation header HDR of a packet PKT as follows. The HDR includes (s_{m_j}, m_j, s, $numAGG$, $aggEntry_1$, $aggEtnry_2$, $\cdots$, $aggEntry_{numAgg}$), where s is the sender address, m_j is the multicast group address. Note that the sender address is different from the source address. If $HDR.numAGG = 0$ i.e., no aggregation entry is included (the packet is delivered through a backup path), it ignores. Otherwise, it performs forwarding table update for each aggregation entry $aggEntry_k = (s_k, d_k, next\_hop_k, seq(k))$ in HDR. If $n \neq d_k$ (i.e., the destination), it looks up the forwarding F for (s_k, d_k, $HDR.m_j$). With found entry, say f_k, k_i updates the entry as follows:

1. if $next\_hop_k = n$ (i.e., a new assign): If $f_k.seq < seq(k)$, it refreshes the next hop to d_k from the local TRN table. It sets $f_k.flag$ and $f_k.agg$ and updates $f_k.seq$ and $f_k.t$, if $f_k.seq \leq seq(k)$. Otherwise, ignore.

2. if $next\_hop_k \neq n$ and $f_k.seq \neq seq(k)$: With $f_k.seq < seq(k)$, it removes the entry, if $seq(k) \geq f_k.seq + r$.

Otherwise, it unsets $f_k.agg$. If $f_k.seq > seq(k)$, ignore. A node removes the entry, if it is not selected as a next hop more than r-1 times. Thus, it allows maximum r duplicate paths (see Fig. 4).

If no entry is found and $next\_hop_k = n$, it creates the entry f_k, updates f_k as (1) and sets $f_k.force$ = TRUE. All entries not updated for T_{update} will be removed from F. The node n processes all entries in HDR, thus, the processing overhead will be $O(|F| * numAGG)$.

If a team leader trn_i sees a new packet, it initiates the intra-team forwarding to be shown in next section. And it forwards the packet to connected leaders in the multicast mesh structure. However, it does not forward a packet to a leader v that trn_i can assure that v has received the packet. For example, M2, in the Fig. 2 (a), excludes S from its designated destination nodes for the packet. If M2 receives a packet from M4 not from S, it excludes M4 and S. A node finds the sender leader by examining aggregation header. When trn_i transmits a packet to other leaders, it initiates the aggregation for destinations.

Packet forwarding process

To decide whether an intermediate node n should relay a packet or not, n examines the forwarding table and the data cache. To differentiate duplicate packets, each node maintains data cache including (the source address s_{m_j}, data sequence number $seqNo$, a multicast group address m_j). A data will be forwarded by n, if (1) n has at least one entry f_k with $f_k.force$ = TRUE and $f_k.m_j = m_j$ or (2) n has at least one entry f_k with $f_k.flag$ = TRUE and $f_k.m_j = m_j$ and receives the new packet. Otherwise, the packet will not be forwarded. When node n relays the packet, it creates the aggregation entries. For each f_k in the forwarding table s.t., $f_k.agg$ = TRUE (i.e., n is in primary path between s_k and d_k) and $f_k.m_j = m_j$, it creates an aggregation entry $aggEntry_k = (f_k.s,\ f_k.d,\ f_k.p_n,\ f_k.seq)$. And it adds the aggregation header $HDR = (s_{m_j},\ m_j,\ s,\ numAGG,\ aggEntry_1,\ aggEtnry_2,\ \cdots,\ aggEntry_{numAgg})$ to the packet and relays the packet. s is the sender address of the packet. After transmitting a packet, n unsets $force$ flag of each entry in F.

Note that the underlying routing protocol DSDV guarantees (within realistic assumptions) a loop-free route discovery. With distance vector routing, a path between two nodes can be dynamic. Using our forwarding flag scheme, we keep the old paths as backup paths, where the maximum number of backup paths depends the parameter r. Multi-path neighbor aggregation scheme improves the reliability both by allowing multiple receptions through a mesh structure and multiple forwarding through multi-paths. Without incurring extra overhead, except for storage of forwarding tables, the forwarding node protocol significantly improves the delivery ratio of the neighbor aggregation mechanism.

4.3 Intra-Team Membership Maintenance and Data Forwarding

There are numerous options to manage the membership and propagate a data within a team. As a potential scheme, we can deploy ODMRP within a team where the leader node periodically floods Join Query packet to the entire team and a member sends a Join Reply back to the leader.

We use a simple approach to handle intra-team membership. This is warranted by the fact that within the team, relative mobility is minimal and only short range because of team affinity. To maintain the team, e.g., the leader reselection, team forming and team split/merge, each node is required to periodically exchange some information. In our implementation, each node exchanges local routing table including entries in $\frac{D}{2}$ hops from a node and a leader is selected based on the routing table information. Without deploying explicit membership join/leave messages, nodes can advertise the membership by piggybacking on the routing table update packets. The data is propagated within a team using a “scoped flooding.

The main advantages of flooding are as follows: it is (1) simple; (2)stateless: flooding does not require to save any state information at intermediate nodes for intra-team data forwarding; (3) robust: the packet reception rate using flooding is very high [11].

5. SIMULATION STUDY

In this section, we evaluate the performance of TOM through extensive simulation experiments. As a reference for performance comparison we use ODMRP (On-Demand Multicast Routing Protocol) [18]. This benchmark choice is justified by the fact that ODMRP was shown to outperform most of the existing ad hoc multicast schemes such as CAMP [16], AMRoute [7] and ARMIS [25] in mobile scenarios [19].

Our performance metrics are as follows:

- delivery ratio: The ratio of the number of delivered packets to each member versus the number of supposedly received packets by each member. Delivery ratio is calculated as follows: $(\sum_{i=1}^{N^g}((\sum_{j=1}^{N_i^m} Recv_{cnt}^j)/(Sent_{cnt}^i * N_i^m)))/N^g$, where N^g is the number of multicast group, N_i^m is the number of members of multicast group i, $Recv_{cnt}^j$ is the number of arrived multicast packets at each member, and $Sent_{cnt}^i$ is the number of sent from the sources of each multicast group
- forwarding overhead: the total number of forwarded data packets versus the total number of delivered packets to members.
- packet latency: the average end-to-end delay of a multicast packet to each member

5.1 Simulation Environments

We use QualNet [1] simulator, a successor of GloMoSim [24]. It provides a detailed and accurate model of the MAC and Channel and routing protocols. We use default parameters provided by QualNet. In our simulation, each source generates data in a CBR (Constant Bit Rate) fashion with UDP (User Datagram Protocol). We use IEEE 802.11 DCF MAC and two-ray ground path-loss model for the Channel. The transmission range of each node is 376m and bandwidth of the device is 2Mbits/sec.

In the network, 1000 nodes are uniformly placed within 6000 x 6000 m^2 terrain. We divide the network into 36 groups where each group has the same group mobility following "Reference Point Group Mobility (RPGM)" model [5]. In RPGM model, each team moves with random vector (speed, direction) and each node in the team randomly moves around the reference point. Except for the mobility study, for all simulations, each team moves with 10 m/s speed with 10 seconds pause time. We assume that the whole group joins a multicast group if a node in the group joins i.e., a group defines a team if it subscribes a multicast group. Thus, maximally 36 teams can exist in the network. The average number of neighbors for each node is 10 and the scope of a team is four. For maintaining the routing structures, ODMRP uses 2 seconds interval for each Join Query and TOM uses 1 second interval for TRN table update. To maintain a team i.e., for a cluster management, each node periodically broadcast its local routing table at every 5 seconds. In our simulation study, we omit the team discovery procedure. We assume that a team is pre-fixed for the simplicity of the evaluation.

For each scenario, multiple runs with different seeds are conducted and the result is averaged over those runs. Each simulation executes for 200 seconds with randomly chosen multicast source(s) and destination team(s). The source sends out four packets every second with 512 bytes packet size as default.

TOM, as default, uses a multicast mesh structure with $m = 3$ and MPNA scheme with $r = 2$ and $I_{update} = 0.25$ seconds.

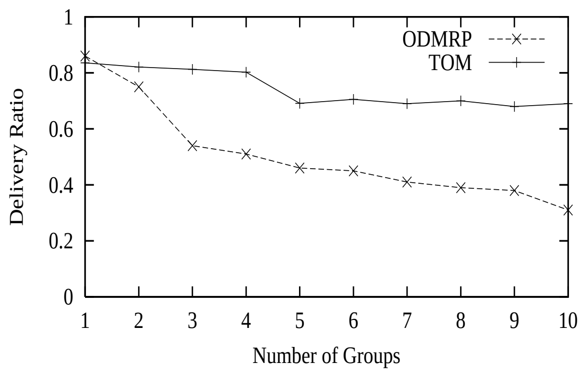

Figure 5: Delivery Ratio v.s. Group Number

5.2 Study on Scalability

One of our main contributions of TOM is the scalability as the group size and number, and network size increases. To show the advantage of TOM compared to traditional *flat* multicast protocols, we examine the throughput changes of TOM over different group number and size compared to those of ODMRP, a representative *flat* MANET multicast protocol. By deploying a large number of nodes (we use 1000 nodes through our simulation), we implicitly show the scalability of TOM with the large number of nodes. To test the scalability with the group number, we increase the number of multicast group(s) from 1 to 10 where each group has five subscribed teams with a single source. For a group size test, we fix the group number and the source number

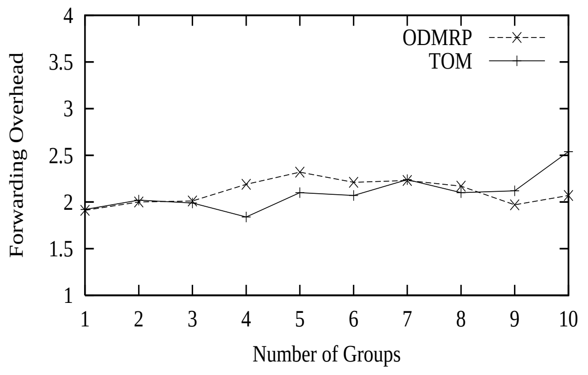

Figure 6: Forwarding Overhead v.s. Group Number

to 1 and increase the number of subscribed teams from 1 to 10.

Fig. 5 and 6 show the delivery ratio and forwarding overhead of TOM compared to those of ODMRP with variable group sizes. The forwarding overhead of both TOM and ODMRP slightly grows as the group number increases; because the network becomes more congested and thus, the delivery ratio degrades. Notably, the delivery ratio of TOM is fairly stable in spite of the increase of offered load. Since TOM does not introduce major control overhead as the group size or number increases, it keeps the network status pretty stable. On the other hand, the performance of ODMRP significantly degrades as the group number increases. As ODMRP applies separate Join Query flooding for each group, the control overhead of ODMRP proportionally increases to the number of group. Thus, ODMRP suffers from heavier load due to the increase of data packets as well as Join Query flood packets as the group number increases. Those results clearly demonstrate the scalability of TOM as the group number increases.

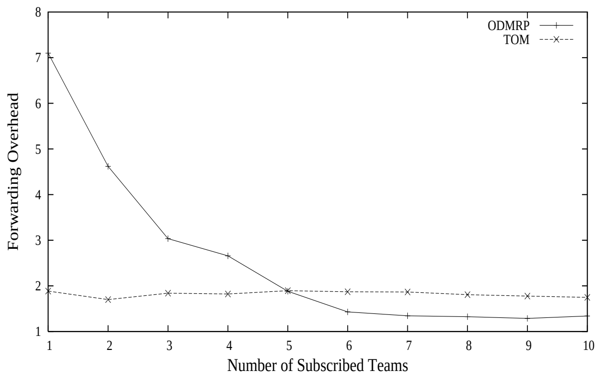

Figure 7: Forwarding v.s. Group Size

Fig. 7 illustrates the forwarding overhead of both schemes versus the group size. Remarkably, in spite of intra-team flooding overhead, the overhead of TOM is comparable to that of ODMRP. More importantly, the overhead of TOM keeps stable. Note that, if we apply an efficient flooding scheme or ODMRP for the intra-team data forwarding as mentioned earlier, the overhead of TOM can be further reduced. On the other hand, the forwarding overhead of

ODMRP is closely related to the group size and actually grows as the group size becomes smaller. Since ODMRP periodically floods a data packet with Join Query message i.e., ODMRP piggybacks the Join Query information on the data packet periodically to update the membership information, the total number of forwarded data packets is dominated by the flooding packets. Thus, the forwarding overhead decreases as the number of members delivering the packet increases. Note that we omit the comparison in terms of delivery ratio with the group size since the next simulation study implicitly shows the result.

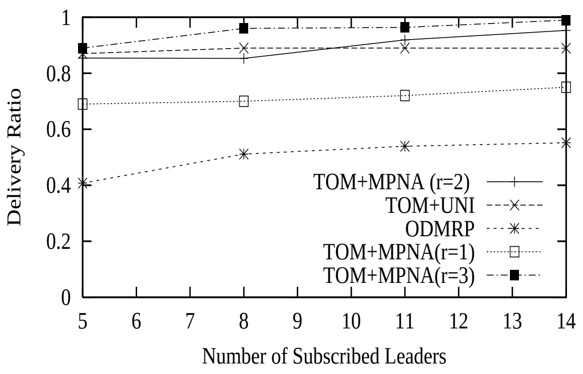

Figure 8: Delivery Ratio of Each Forwarding Scheme

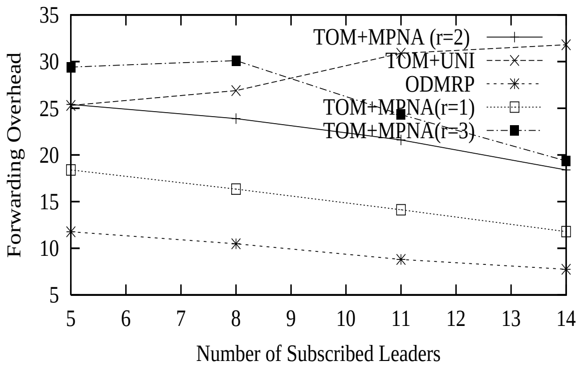

Figure 9: Forwarding Overhead of Each Forwarding Scheme

5.3 Investigation on Forwarding Mechanisms

In this simulation, we investigate the performance of inter-team forwarding mechanisms: (1) separate unicast tunneling; and (2) multi-path neighbor aggregation technique with $r = 1$ i.e., a single path, $r = 2$ and $r = 3$. As we study the throughput of first-tier nodes, we omit the intra-team forwarding in this experiment. Thus, only team leaders become member nodes of a multicast group. And we use fixed team leaders randomly chosen at the initialization of each simulation run. We examine the performance of ODMRP over subscribed leaders, as a reference.

We use a multicast group forming the multicast mesh (m = 3) with a single source and variable number of members from 5 to 14.

In Fig. 8 and 9, we can observe four important facts. First, the delivery ratio of the single-path broadcast schemes used by ODMRP and MPNA with r =1 are remarkably low compared to that of unicast tunneling. Still, the redundant packet transmission in the multicast mesh significantly improves the reliability i.e., TOM+MPNA (r=1) performs far better than ODMRP. Secondly, the multi-path mechanism considerably enhances the throughput. By adding one more path i.e., r =2, the performance of TOM is improved more than 20%. As the throughput difference between r=2 and r=3 is not significant, we recommend to use r = 2 for MPNA technique. Thirdly, the forwarding overhead of unicast increases as the number of connections increases. On the other hand, broadcast mechanisms reduce the overhead and efficiently forward a packet by eliminating unnecessary rebraodcasts of the same packet. Thus, broadcast mechanisms are much more scalable than unicast tunneling with group size. Lastly, our virtual forwarding structure becomes more robust and efficient with the group size. The forwarding overhead of MPNA scheme degrades but the reliability of it increases as the group size grows.

Note that, to collect the forwarding overhead of ODMRP in this simulation study, we omit the number of periodic data flooding packets (i.e., Join Query flooding packets). Thus, the forwarding overhead of ODMRP represents the number of re-broadcasts to deliver a data packet through the underlying ODMRP mesh structure. With the single path scheme and underlying forwarding mesh structure, ODMRP is more efficient than TOM in terms of the forwarding overhead. We consider the overhead of ODMRP as the lower bound to propagate a data within a multicast mesh structure. Considering that each team has many members, TOM does not significantly increase the forwarding overhead even though it applies multiple paths and redundant transmissions.

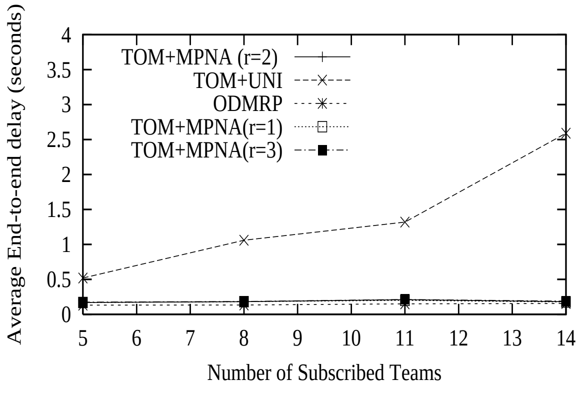

Figure 10: End-to-end Delay using Each Forwarding Scheme

Fig. 10 shows the packet delivery latency of each protocol. Now, all nodes in a team join the multicast group and intra-team data forwarding is used. Remarkably, unicast tunneling suffers from the higher latency as the network becomes more overloaded. While, broadcast mechanisms show very low latency in spite of broadcast jitter delay at each forwarding node. Note that each forwarding node waits a random time before transmitting a broadcast packet to prevent concurrent transmissions among neighbors (we use 10 milliseconds for the broadcast jitter).

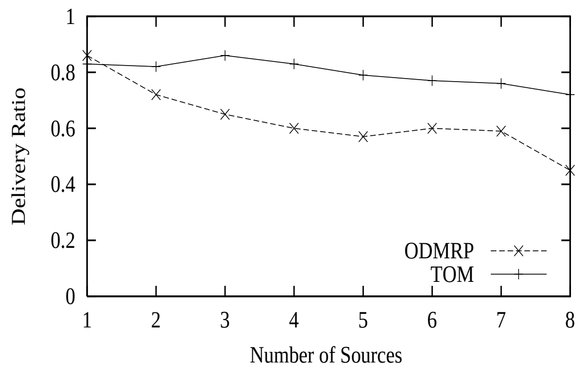

Figure 11: Delivery Ratio v.s. Number of Sources

5.4 Study with Increasing Number of Sources

As we mentioned earlier, TOM is designed for scenarios with multiple sources. In this experiment, we evaluate the performance of TOM with variable number of sources. For the simulation, we use one multicast group and nine randomly selected teams. We increase the number of sources randomly chosen among members from 1 to 8.

Fig. 11 shows that the delivery ratio of TOM keeps stable as we increase the number of sources. Because of the mesh structure not having any dependency on the source node, TOM performs well regardless of the number of sources. On the other hand, the performance of ODMRP degrades as the number of sources increases due to the increase of control overhead by Join Query floods.

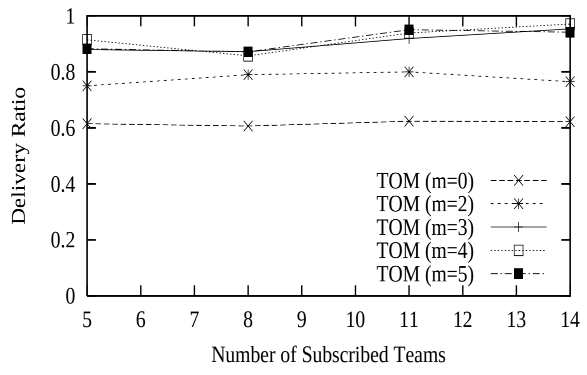

Figure 12: Delivery Ratio v.s. Mesh Degree

5.5 Impact of Mesh Degree on Performance

Intuitively, the packet delivery ratio of a mesh structure will be enhanced as m increases unless the network is congested. In this simulation, we want to investigate the impact of redundancy degree m on the delivery ratio and forwarding overhead. As a reference, we build a 1-level multicast tree with $m = 0$.

For the simulation, we use a multicast group with a single source. We increase the number of subscribed teams from 5 to 14.

In the results, Fig. 12 and 13, we can observe two major performance improvements between $m = 0$ and $m = 2$ and between $m = 2$ and $m = 3$. The results clearly demonstrate the benefit from the path redundancy created by using the mesh structure. However, a mesh structure with a large redundancy factor more than three does not significantly improve the throughput. Notably, with a mesh with m =5 suffers and performs actually worse than m =4 case due to too heavy forwarding overhead. Empirically, we recommend m =3 to maximize the throughput of the proposed mesh structure.

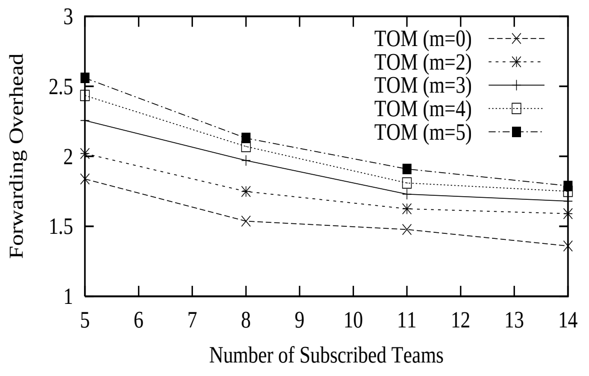

Figure 13: Forwarding Overhead v.s. Mesh Degree

5.6 Effect of Team Mobility

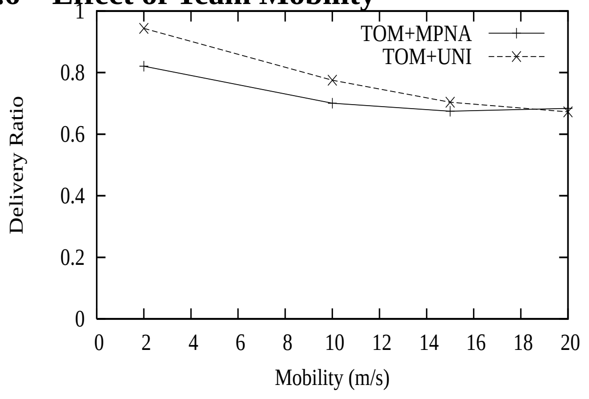

Figure 14: Delivery Ratio v.s. Mobility

As we target a reliable data forwarding platform among team leaders in a highly dynamic mobile network, we evaluate the performance degradation of TOM as the mobility increases. We increase the group mobility from 2 m/s to 20 m/s with 10 seconds pause time with a single source (4pkts/sec data rate) and seven randomly selected teams. We use $m = 3$ for both MPNA and unicast tunneling schemes and $r = 2$ for MPNA scheme.

Fig. 14 shows the delivery ratio with the changing mobility values. In both schemes, the delivery ratio degrades as the mobility increases. However, the extent of the performance damage by mobility of each protocol is different. The delivery ratio of unicast tunneling sharply drops mainly due to the increase of route breakages as the node mobility increases. However, MPNA, by deploying multiple paths, manages the frequent path breakages and thus improves the reliability. As the result shows, MPNA becomes more effective as the mobility gets higher.

6. CONCLUSION

In the paper, we proposed a two-tier hierarchical multicasting protocol exploiting the affinity team model where there is a set of nodes that share the motion affinity (and possibly other interests) and are placed near each other. Our proposed idea, named TOM, contributed as follows: (1) by reducing the number of visible members from outside, TOM considerably reduces the complexity and overhead of a multicasting protocol; (2) TOM identified and corrected the low packet delivery ratio in the large-scale network, which should be addressed to develop a scalable MANET protocol; (3) TOM developed a multicast mesh structure and multi-path neighbor aggregation technique to improve the reliability; (4) through extensive study, TOM showed the scalability, reliability, flexibility and efficiency.

For our future work, a growing demand for multicast QoS, we believe that TOM can be successfully extended to support a reliable transport protocol, multicast congestion control and real-time multicast applications.

APPENDIX

A. THEOREMS

Theorem 1. *If all v ($v \neq$ the root vertex r) has a link with w such that $seq_v > seq_w$, then the undirected graph G constructed following our Join Procedure algorithm is connected.*

Proof. Let $V = \{v_1, v_2, \cdots, v_n\}$ sorted by the sequence number be a set of nodes in G. We prove the theorem using the contradiction. With the assumption that *all v* ($v \neq$ the root vertex r) has a link with w such that $seq_v > seq_w$, if G is not connected, then there is node v_i and v_j that do not have a link between them. Let the set of connected nodes to v_i be V_{v_i} and the set of connected nodes to v_j be V_{v_j}. Since v_i and v_j should not be connected each other, V_{v_i} and V_{v_j} are disjoint. Let r_i be the node who has the lowest sequence number in V_{v_i} and r_j be the node who has the lowest sequence number in V_{v_j}. To satisfy the assumption that only root does not have a parent node, r_i and r_j should be both r, but it is not possible. Thus, v_i and v_j cannot be unreachable from each other. □

Theorem 2. *There is at least one node v who is connected to more than two parents or has less than m links in G constructed following our Join Procedure if $m \geq 2$.*

Proof. If all nodes have only one parent, the graph should be a string topology. With a string topology, at least two (start node and end node of the string) should have one link. If each node has more than two links, at least one node should have two parents because it is undirected graph. Thus, it contradicts the assumption. There is at least one node who has more than two parents or less than two links. □

B. REFERENCES

[1] Scalable networks, http://www.scalble-solutions.com.

[2] B. An and S. Papavassillou. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks. *International Journal of Network Management*, 2001.

[3] T. Ballardie, B. Cain, and Z. Xhang. Core based tree (cbt version 3) multicast routing protocol specification. *Internet draft*, March 1998.

[4] T. Ballardie, P. Francis, and J. Crowcroft. Core-based trees (cbt): An architecture for scalable inter-domain multicast routing. *ACM SIGCOMM*, 1993.

[5] G. Bianchi, X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. *Proceedings of ACM/IEEE MSWiM'99*, 1999.

[6] B.McDonald and F.Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, vol. 17, Aug. 1999.

[7] E. Bommaiah, M. Liu, A. McAuley, and R. Talpade. AMRoute: Ad-hoc Multicast Routing protocol. *Internet-draft, draft-talpade-manet-amroute-00.txt*, 1998.

[8] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking*, 2002.

[9] D. S. J. DeCouto, D. Aguayo, B. A. Chambers, and R. Morris. Effects of loss rate on ad hoc wireless routing. *technical report MIT-LCS-TR-836*, March 2002.

[10] M. Gerla, X. Hong, and G. Pei. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. *Proceedings of IEEE/ACM MobiHOC*, 2000.

[11] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. 1999.

[12] X. Hong and M. Gerla. Dynamic group discovery and routing in ad hoc networks. *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop*, 2002.

[13] X. Hong, M. Gerla, Y. Yi, K. Xu, and T. Kwon. Scalable ad hoc routing in large, dense wireless networks using clustering and landmarks. *IEEE ICC*, 2002.

[14] R. Hutchins and E. Zegura. Measurement from a campus wireless network. *ICC*, 2002.

[15] L. Ji and M. S. Corson. Differential destination multicast-a manet multicast routing protocol for small groups. *INFOCOM*, 2001.

[16] J.J.Garcia-Luna-Aceves and E. L. Madruga. A multicast routing protocol for ad-hoc networks. *IEEE INFOCOM*, 1999.