OS-multicast: On-demand Situation-aware Multicasting in Disruption Tolerant Networks

Qing Ye, Liang Cheng, Mooi Choo Chuah, and Brian D. Davison
Department of Computer Science and Engineering, Lehigh University
Bethlehem, PA 18015, USA
{qiy3, cheng, chuah, davison}@cse.lehigh.edu

***Abstract*—Disruption Tolerant Networks (DTNs) are emerging solutions to networks that experience frequent network partitions and large end-to-end delays. In this paper, we study how to provide high-performance multicasting service in DTNs. We develop a multicasting mechanism based on on-demand path discovery and overall situation awareness of link availability (OS-multicast) to address the challenges of opportunistic link connectivities in DTNs. Simulation results show that OS-multicast can achieve a better message delivery ratio than existing approaches, e.g. DTBR (a dynamic tree-based routing), with similar delay performance. OS-multicast also achieves better efficiency performance when the probability of link unavailability is high and the duration of link downtime is large.**

Keywords-Disruption tolerant networks; multicasting

I. INTRODUCTION

Data communication challenges exist in network scenarios where an instantaneous end-to-end path between a source and destination may not exist because links between nodes may be opportunistic or predictably/periodically connected. There is ongoing research [2, 3, 4, 5] on disruption tolerant networks (DTNs) that addresses such challenges. DTNs have a broad range of potential applications such as military battlefields [7], deep-space communications [8], habitat monitoring [9], and Internet access in rural areas [10].

Many DTN applications need multicast service. For example, in military battlefields, it is vital to quickly and reliably transmit orders from a command center to a group of field commanders. It is also helpful to share information of surrounding environments among different squads of soldiers. However, traditional multicast methods proposed for the Internet (e.g., MOSPF [11] and DVMRP [12]) or mobile ad hoc networks (e.g., AMRoute [13] and ODMRP [14]) are not suitable for DTNs, due to the challenge of frequent network partitions. Firstly, it is difficult to maintain the connectivity of a source-rooted multicast tree (or mesh) during the lifetime of a multicast session. Secondly, data transmissions suffer from large end-to-end delays along the tree because of the repeated disruptions caused by periodically broken branches. Thirdly, the traditional approaches may fail to deliver a message when the possibility of link unavailability becomes high (e.g. ~80%). In this paper, we address these issues and focus on studying how to provide high-performance multicasting in DTNs in terms of high delivery ratio and short delay.

We investigate three multicasting methods for DTNs, which employ unicast-based, static tree-based and dynamic tree-based strategies respectively. We propose an on-demand situation-aware multicast (OS-multicast) approach, which is a new dynamic tree-based method that integrates DTN multicasting with the situation discovery by the underlying network layer. DTBR [2] is another dynamic tree-based multicasting approach for DTNs. Simulation results show that OS-multicast can achieve smaller delays and better message delivery ratios than DTBR. OS-multicast also achieves higher efficiency when the probability of link unavailability is high and the duration of link downtime is large.

The rest of the paper is organized as follows. Section II presents descriptions of the network model and multicast model of DTNs. Section III explains the basic DTN multicast approaches and the OS-multicast algorithm. Performance evaluations are illustrated in Section IV. Section V summarizes our contributions.

II. SYSTEM MODELS

A. Network Model

We view a DTN as an overlay built upon underlying networks, such as wireless ad hoc networks. Its network architecture is based on the asynchronous message (called bundle) forwarding paradigm presented in [1]. Only those nodes that implement the DTN functionalities to send and receive bundles are DTN nodes, while the others are denoted as normal nodes. A DTN link may contain several underlying links in multiple hops. Fig. 1 depicts a simple example.

In the DTN layer, bundles are transmitted in a hop-by-hop store-and-forward manner. Each DTN node has finite-size buffers for bundle acceptance and bundle custody. The details of custodian transfer are discussed in [4]. They are out of the scope of this study. Normally, the underlying network layer provides unicast routing capability to forward a bundle from one DTN node to another. The multicast service discussed in this research is only implemented in the DTN overlay. More details about our DTN architecture can be found in [5].

B. Multicasting Model

Multicast in DTNs is defined as the one-to-many or many-to-many bundle transmissions among a group of DTN nodes. Each DTN node is associated with a DTN *name* that potentially permits late binding to its underlying network address. The address translation between the DTN *name* and underlying network address is done by DTN routing agent. A multicast

source uses a group name or a explicit list of the names of DTN receivers as the destination address for transferring bundles. The details of multicast membership management are discussed in Section III.

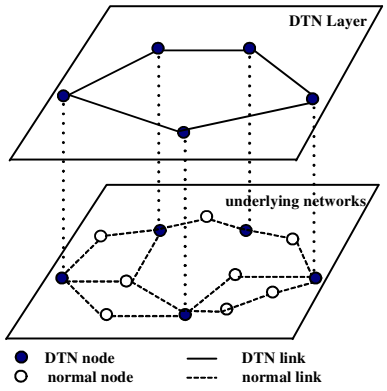

Figure 1. A simple example of DTN networks.

III. Multicast Routing Approaches and Algorithms

A. Situation Awareness

As we mentioned before, DTNs suffer from frequent network partitions. The uncertainty of network topology and the dynamic changes of the link state make the maintenance of the multicast tree in DTNs challenging. The performance of different multicast approaches depends on the knowledge of network conditions discovered in DTN nodes. Therefore, situational awareness can be applied to help DTN nodes control the message forwarding behavior and discover different message delivery paths, based on the policies and network conditions at different times.

Situational awareness is achieved by making the multicast implementation collaborate with the routing methods in the underlying networks. We assume that before the DTN overlay starts, the underlying network has already been operating for a relatively long time. Periodically, a DTN routing agent will send *situation_req* message to trigger its underlying routing agent to collect the current network conditions, such as the presently available outgoing links and the discovered paths from the current DTN node to the destinations. This requirement could be fulfilled by some source routing approaches such as DSR [6]. The underlying routing agent then answers *situation_resp* message with all the detected information back to the DTN routing agent. Thus, a knowledge base of the link state and network topology is constructed in each DTN node. Both *situation_req* and *situation_resp* messages are system messages transmitted inside a node.

B. Multicast Approaches

We study three approaches for supporting multicast communications in a DTN. A simple example of these approaches is illustrated in Fig.2.

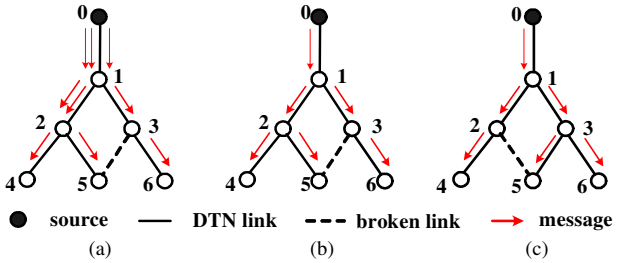

Figure 2. Multicast approaches in DTN. (a) U-multicast; (b) ST-multicast; (c) OS-multicast: when link 2→5 is unavailable and link 3→5 becomes available, node 3 will take advantage of the current available link immediately.

- ***Unicast-based multicast*** (***U-multicast***). This approach implements multicast service by using multiple source-to-destination unicast data transfer. In U-multicast, the source always tries to set up an end-to-end path to each destination and sends a copy of the bundle along the currently available shortest path. If there is no such path for a destination due to a network partition, the source will hold the bundle in its buffer and retransmit it once the destination is connected.
- ***Static Tree-based multicast*** (***ST-multicast***). In ST-multicast, a source-rooted multicast tree is constructed at initialization by querying the shortest paths to reach each destination from its knowledge base. Then the source keeps sending bundles along this static tree and requires the intermediate DTN nodes with multiple downstream neighbors to duplicate the messages. Compare to U-multicast, each bundle is pushed to the DTN nodes that are nearest to the destinations using the available links, even if the end-to-end paths in the static tree are unavailable. Once the broken downstream links become available, the buffered messages will be forwarded further.
- ***On-demand Situation-aware multicast*** (***OS-multicast***). Unlike the above two methods, OS-multicast is a dynamic tree-based multicast approach. A unique multicast tree is built up for each bundle and the tree varies according to the changes of the network at each intermediate DTN node. First, a source-rooted tree is constructed in the similar way as ST-multicast. When a DTN node receives a bundle, it will dynamically rebuild the tree rooted at itself to all the destinations based on its knowledge of the current network conditions. If there is a newly available path to a destination, which was not discovered by the upstream DTN nodes, this node will immediately take advantage of that fresh information and send the bundle out.

C. OS-multicast Algorithm

In this section, we introduce the details of the proposed dynamic tree-based OS-multicast algorithm.

a) Membership management

When a DTN node intends to join a multicast group, it

registers with its membership period by explicitly sending a *GROUP_JOIN* message. For example, node *i* wants to join the multicast service during the period [t_{si}, t_{ei}], with the start-time t_{si} and the end-time t_{ei} of its membership. When the multicasting source is informed by the *GROUP_JOIN* message, it puts the membership information into a *membership_list*, denoted as L_M in each bundle. For every bundle received, generated or retransmitted at time t, a DTN node will check the validity of each receiver in L_M. If the membership of a receiver has expired ($t > t_{ei}$) or was not activated ($t < t_{si}$), then it will not forward the bundle to that receiver. A receiver is called a valid receiver only if it owns valid membership for that bundle. This membership management method conforms to the TM semantic model proposed in [2] with an explicit receiver list known at the source.

b) Bundle storage

Each DTN node has a local storage with finite size. A received bundle will be saved in the buffer until eliminated due to buffer overflow or successfully forwarded toward all valid receivers based on the bundle acceptance policy. For example, when the buffer is full, the bundle at the head of the buffer will be dumped first with the arrival of a new bundle.

c) Forwarding state maintenance

A forwarding state is associated with each bundle. There is an *upstream_list* (called L_U) maintained in the bundle and a *pending_list* (called L_P) maintained by the DTN node. When a bundle arrives, a DTN node creates L_P for that bundle by copying L_M information from the bundle. L_P is periodically checked to remove information corresponding to those receivers whose memberships have expired. When node *A* wants to forward the bundle to a downstream node *B* to reach a valid receiver *C*, it first checks if $B \in L_U$ to avoid the redundancy. Then it removes *C* from L_P. Once L_P is empty, the bundle is then removed from the local buffer of node *A*.

d) Message forwarding

When a bundle is generated, the source queries its knowledge base to retrieve all the discovered paths to the receivers. Then the source combines those paths to be a multicast mesh that covers all the valid receivers. We denote it as a *static-mesh*. The source then filters this mesh by deleting those currently unavailable outgoing links reported by the underlying routing agent. We denote the result as a *dynamic-mesh*. Based on the *dynamic-mesh*, a source-rooted shortest-path multicast tree is built and the source forwards the bundle to all its available downstream DTN neighbors, carrying the *static-mesh* information in the bundle.

Upon receiving a bundle, a DTN node *X* will first query its knowledge base to find all the possible paths to the receivers. Then it combines the *static-mesh* in the bundle with the query results to be a new static-mesh. A *dynamic-mesh* is then constructed in the same way as what the source has done. Then the DTN node *X* re-computes the shortest-path multicast tree by taking itself as the root to all the receivers. The new *static-mesh* is put into the bundle again and forwarded further to downstream neighbors. With the knowledge propagation and the dynamic decision made by each intermediate DTN node, eventually the bundle will arrive at those receivers in L_M. Note that the same bundle will not be forwarded more than once at each node to avoid introducing redundant traffic into the networks.

e) Bundle retransmission

A DTN node periodically checks its local storage to see if there is any opportunity to forward the buffered bundles further. It uses the current *dynamic_mesh* to find out if there is a chance to forward the bundle to a receiver in L_P. If so, the bundle is forwarded using this available forwarding opportunity and the covered receiver is removed from L_P. To reduce the overhead of the OS-multicast, there is an upper-bound R_{upper} to limit the maximum retransmission times for each receiver. If the retransmission to receiver *D* fails more than R_{upper} times, *D* would be removed from L_P. Once L_P is empty, the bundle will be deleted from the local buffer.

D. Comparisons between OS-multicast with DTBR

DTBR [2] is another dynamic tree-based multicasting algorithm designed for DTNs. Similar to OS-multicast, DTBR also requires the intermediate DTN nodes to re-build a multicast tree. However, in each step of the bundle transmission, the upstream node will assign the receiver list for its downstream neighbors based on its local view of the network conditions. The downstream nodes are required to forward bundles only to the receivers in the list, even if a new path to another receiver (not in the list) is discovered. Fig. 3 depicts this issue.

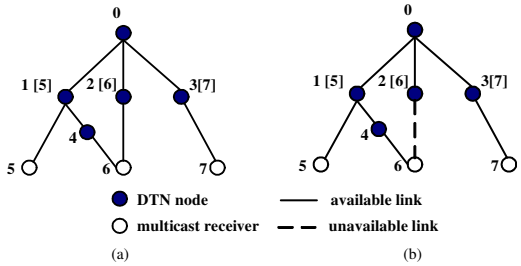

Figure 3. (a) At time t_0, the source (node 0) computes the multicast tree and decides that the receiver list for node 1 is {5}, for node 2 is {6}, and for node 3 is {7}. (b) At time $t_1 > t_0$, link 2→6 is down. Node 1 cannot forward the bundle to node 6 even if there is an available path 1→4→6 just because node 6 is not in its receiver list. Before the source detects this problem, it will keep sending bundles to node 2 and ignore the other better opportunity.

This issue is solved in OS-multicast because each intermediate node has an equal chance to decide the receiver list. If a link towards a receiver becomes available, the DTN nodes which detect that will immediately take advantage of this new opportunity. We call this greedy nature of OS-multicast as the *first availability property*. This property guarantees that the bundle will be delivered to the receivers as soon as possible and the opportunistic links will be utilized.

DTBR assumes that each node has complete knowledge or the summary of the link states in the network. However, this is

hard to be satisfied in most practical applications. In our design of OS-multicast, we use situational awareness with the help of underlying routing methods to collect the information of network conditions, such as the currently available outgoing links and the possible paths to destinations.

IV. Performance Evaluation

To evaluate the performance of different multicast algorithms in this study, we implemented U-multicast, ST-multicast, OS-multicast and DTBR in the *ns2* simulator. Our performance metrics include: *i*) *message delivery ratio*, which is defined as the number of unique multicast bundles successfully arrived at all the receivers over the total number of bundles which are expected to be received; *ii*) *efficiency*, which is the ratio between the unique bundles received by the receivers and the total traffic generated in the networks; and *iii*) *average message delay*, which is the average of end-to-end bundle transmission delays for each algorithm (we observe similar delay performance results when we use the metric of median delay). To make our comparison fair, we only calculate the delay of those bundles received by both multicasting methods because each method has different delivery capability.

All simulations are based on a network of 25 nodes deployed in a 1000×1000 area. The topology is a grid network. Fifteen DTN nodes are randomly selected to be the DTN nodes while the remaining are normal nodes. Multicasting algorithms are only implemented in those DTN nodes. DSR is used as the routing approach for the underlying ad hoc networks. And situational awareness is achieved by the communication between DTN multicasting agent and the DSR routing agent. The MAC layer is IEEE 802.11 with radio transmission range of 250 meters.

We study one multicast session in the DTN overlay. Node 0 is fixed to be the source and 5 DTN nodes are randomly chosen to be the receivers. The message sending rate is 1 bundle per 2 seconds with the bundle size of 512 bytes. Each DTN node can maximally keep 100 bundles in its local buffer. At every 5 seconds, each DTN node will query the underlying routing agent to find if there is any available opportunistic link to forward the buffered bundle to the destinations. If so a copy of the bundle will be forwarded. We study the performance of different multicasting algorithms by varying the percentage of link unavailability of each link from 10% to 90%.

Fig. 4 shows the result of the message delivery ratios of U-multicast, ST-multicast, OS-multicast and DTBR. We observe that *i*) the delivery ratio decreases for all the algorithms when the downtime of links becomes large, i.e., the network is more easily partitioned; *ii*) OS-multicast can always achieve the best performance among all the algorithms; and *iii*) U-multicast performs better than ST-multicast when the downtime is small because it basically tries to forward bundles in a multicast mesh than a tree. For each bundle transmitted by each intermediate DTN node, OS-multicast tries to utilize multiple paths to the receivers and take advantage of the currently available opportunistic links to push the data closer to the destinations.

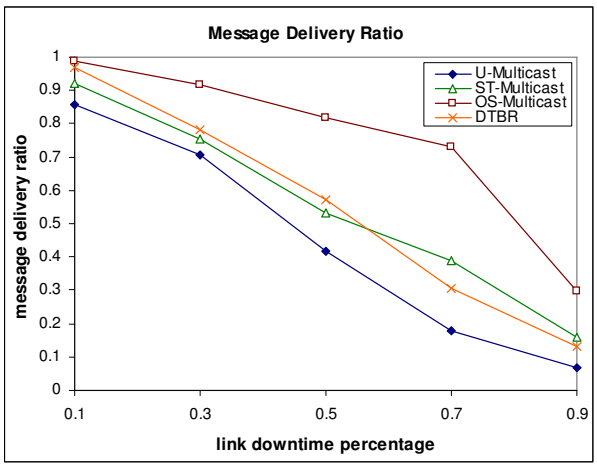

Figure 4. Message delivery ratios of different multicasting algorithms.

The above simulation is conducted by uniformly dividing the downtime of each link into 10 small periods. To better show the performance of OS-multicast, we also test the message delivery ratio performance by dividing the total downtime into 30 periods, i.e., making the link states vary more often and increasing the occurrences of opportunistic links. The result is illustrated in Fig. 5. It is obvious that *i*) all the methods deliver fewer bundles with more frequent link changes, and *ii*) OS-multicast outperforms the other approaches more than the result shown in Fig. 4. This demonstrates that OS-multicast is able to utilize the opportunistic links more effectively.

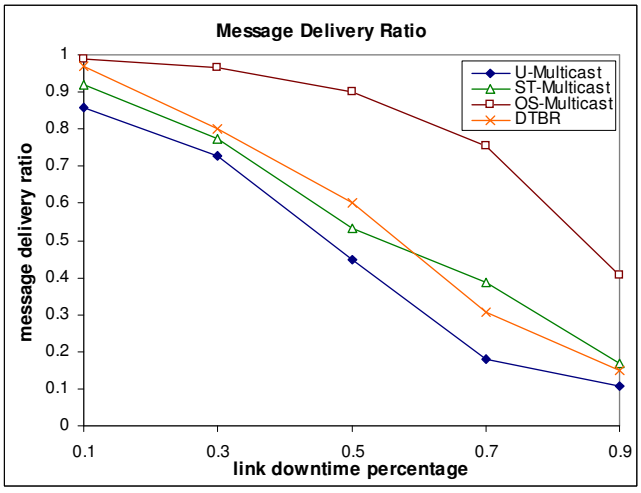

Figure 5. Message delivery ratios of different multicasting algorithms with more occurances of opportunistic links.

The following simulations are all carried out by dividing the downtime of each link into 10 periods. Fig. 6 depicts the median message delay of two dynamic-tree based multicasting approaches: OS-multicast and DTBR. OS-multicast has smaller delays than DTBR due to the issues illustrated in Fig.

3. In DTBR, each node only forwards bundles to the downstream nodes to reach the receivers in its receiver list. However, this receiver list is decided by the upstream node based on its snapshot of the network conditions. In this way, some opportunistic links to the receivers that are not in the list are missed. In contrast, the OS-multicast method always uses all the chances to forward bundles to the destinations. Note that results in [2] already show that DTBR achieves slightly better delay performance than ST-multicast.

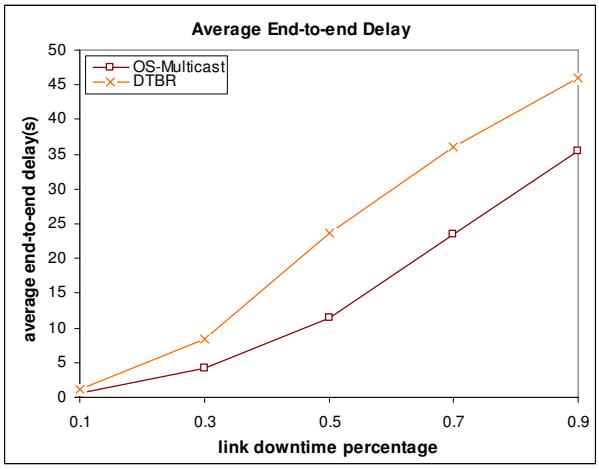

Figure 6. Average message delays between OS-multicast and DTBR.

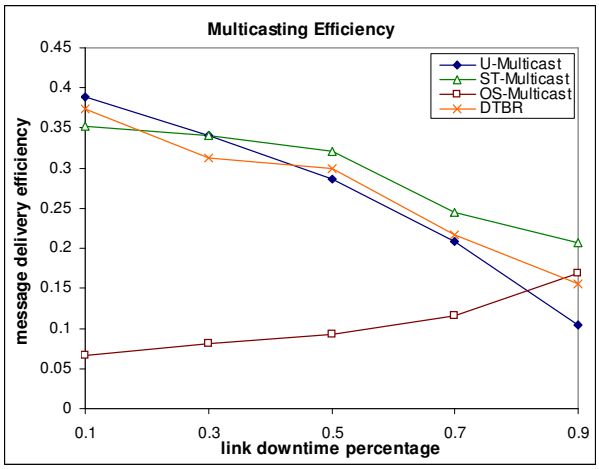

Figure 7. Multicast efficiency of different algorithms.

We also evaluate the efficiency of various multicasting approaches and illustrate the result in Fig. 7. It shows that although OS-multicast has the highest delivery ratio it has the worst efficiency when the link downtime is small. The reason is that a lot of redundant traffic has been introduced by OS-multicast due to its nature of utilizing multiple currently available links. However, when the link downtime percentage is larger than 80% of the total simulation time, its efficiency increases because the chance of redundancy decreases and it is still able to achieve the relatively high delivery ratio as shown in Fig.4. The efficiencies of the other three algorithms decrease while the network connectivity becomes worse.

V. Conclusions

We have developed an on-demand situation-aware multicast (OS-multicast) algorithm, which is able to dynamically adjust multicast routing decisions based on the current network conditions. We compare its performance with those achieved by DTBR, ST-multicast, and U-multicast. Simulation results OS-multicast can achieve better message delivery ratio than existing approaches with similar delay performance. When network connectivity becomes worse due to the high link unavailability, OS-multicast also has better efficiency. We are currently investigating on an improved version of our OS-multicast scheme that can achieve higher efficiency at low/medium link availability.

Acknowledgment

This work has been supported by DARPA under Contract W15P7T-05-C-P413. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor of this work.

References

[1] K. Fall, “A delay-tolerant network architecture for challenged Internets”, *Proceedings of SIGCOMM*’03, August 2003.

[2] W. Zhao, M. Ammar, and E. Zegura, “Multicasting in delay tolerant networks: semantic models and routing algorithms,” in the *Proceeding of Sigcomm Workshop in DTN*, August 2005.

[3] DARPA Disruption Tolerant Networks program http://www.darpa.mil/ato/solicit/dtn/, accessed on Aug. 3rd, 2005.

[4] K. Fall, “Messaging in difficult environments,” Intel Research Berkeley, *IRB-TR*-04-019, Dec. 27, 2004.

[5] M. Chuah, L. Cheng, B. Davison, “Enhanced disruption and fault tolerant network architecture for bundle delivery (EDIFY)”, to appear at *Globecom*, 2005.

[6] D.B. Johnson and D.A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, edited by T. Imielinski and H. Korth, chapter 5, pp.153-181, Kluwer Academic Publishers, 1996.

[7] Rajeswari Malladi and Dharma P. Agrawal, “Current and future applications of mobile and wireless networks”, *Communications of the ACM*, Vol. 45, pp. 144-146, 2002.

[8] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, “Delay-tolerant networking - an approach to interplanetary internet”, *IEEE Communications Magazine*, June 2003.

[9] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, Jerry Zhao, “Habitat monitoring: application driver for wireless communications technology”, in the *Proceeding of ACM SIGCOMM Workshop on Data Communications*, April 2001.

[10] S. Jain, K. Fall, R. Patra, "Routing in a delay tolerant networking", in the *Proceedings of SIGCOMM*’04, Aug./Sep. 2004.

[11] J. Moy, “Multicast extensions to OSPF”, IETF RFC 1584, 1994.

[12] D. Waitzman, C. Partridge and S. Deering, “Distance vector multicast routing protocol (DVMRP)”, *IETF RFC* 1075, 1988.

[13] J. Xie, R.R. Talpade, A. Mcauley, and M.Y. Liu, “AMRoute: ad hoc multicast routing protocol,” *Mobile Networks and Applications*, Vol. 7, Issue 6, pp. 429-439, 2002.

[14] Sang Ho Bae, Sung-Ju Lee, William Su, and Mario Gerla, “The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks”, *IEEE Network*, pp.70–77, January 2000