

Performance Comparison of Multicasting Approaches 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 partitions. In this paper, we study how to provide high-performance multicasting service in DTNs. We first investigate three basic DTN multicast approaches. We then develop a multicast 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. Performance of different DTN multicast approaches are evaluated by simulations, including tests that use trace files of practical DTN experiments. Simulation results show that OS-multicast can achieve a better message delivery ratio than existing approaches with smaller delays in multiple scenarios. OS-multicast also achieves better message delivery efficiency performance when the link unavailability becomes large and the lack of network connectivity becomes severe.

Keywords—disruption tolerant networks; multicasting; performance comparison

I. INTRODUCTION

Recent research in Disruption Tolerant Networks (DTNs) [1, 2, 3, 4] has raised several issues of handling data communication in networks where instantaneous end-to-end paths become unstable and inconsistent. DTNs are featured by frequent network partitions as the result of planned or unplanned link up/down periods between neighboring nodes. Those link layer challenges may be caused by: highly mobile nodes; low network density; a lack of long range radios; certain energy management schemes to put nodes into sleeping modes; or infrastructure disruptions due to environmental disasters, military operations, or terrorist attacks. DTN technologies are viewed as important components to support a broad range of applications such as military battlefields [5], deep-space communications [6], habitat monitoring [7], and Internet access in rural areas [8].

Many DTN applications need multicast service. For example, in military battlefields, it is vital to quickly and reliably deliver orders from a command center to a group of field commanders. It is also helpful to share information regarding surrounding environments among different squads of soldiers. Moreover, multicast could help geographically disconnected rural areas utilize the opportunities of accessing Internet more effectively and efficiently, for applications like

data dissemination, file sharing, etc.

However, traditional multicast methods proposed for the Internet (e.g., MOSPF [9] and DVMRP [10]) or mobile ad hoc networks (e.g., AMRoute [11] and ODMRP [12]) are not suitable for DTNs, due to the challenge of frequent network partitions. Firstly, it is difficult to maintain the connectivity of a multicast structure (tree or mesh) during the lifetime of a multicast session. Secondly, data transmissions would suffer from a large number of failures and large end-to-end delays because of the disruptions caused by repeatedly broken multicast branches. Thirdly, the traditional approaches are designed based on the assumption that the underlying networks are basically connected. In DTNs, the presence of frequent topology variations gives rise to the need to be able to adjust the multicast structure to adapt to the changes of network conditions.

We offer several contributions in this study: i) we investigate three kinds of possible multicast approaches for DTNs, which employ multiple-unicast-based, static-tree-based, and dynamic-tree-based strategies respectively; ii) we propose a new DTN multicast protocol, called OS-Multicast, which is able to dynamically build the multicast tree according to currently available DTN links with the help of situation discovery by the underlying network layer; iii) we then conduct performance comparisons of these approaches by a broad range of sensitivity studies from varying the DTN inter-contact durations to applying the real-world DTN traces, including the Pocket Switch Network (PSN) trace [14] and the DieselNet trace [15]. Simulation results show that OS-multicast can achieve better message delivery ratios and smaller average delays than previous DTN multicasting approaches.

The rest of this paper is organized as follows. We summarize related works in Section II. Section III presents our network model and multicast model of DTNs and introduces the basic ideas of different multicast approaches. Section IV explains the details of our proposed OS-multicast algorithm. The performance comparisons among multicast methods presented in this paper are illustrated in Section V. And finally, Section VI concludes this study.

II. BACKGROUND AND RELATED WORK

The characteristics of occasionally-connected networking in

DTNs could be found in many networks that are subject to disruptions and disconnections. For example, the Interplanetary Internet designed to support deep-space data transmissions is a typical scenario for DTNs [16]. Those wireless links connecting spaceships or space stations may be periodically unavailable. Data communications via satellites among several military bases across different continents is another example as data can be exchanged only when a satellite is visible to a military base. Wireless sensor networks which are deployed in very harsh environments may also suffer from frequent network partitions. Moreover, in the networks consisting of highly mobile vehicles without powerful antennas, the duration of an available link between two moving nodes would be very short.

To overcome the frequent disruptions of end-to-end paths, a virtual message switching scheme is proposed in DTN routing designs. Basically, a DTN node is assumed to have finite local buffer to hold messages that it generates and receives. Asynchronous messages (called bundles) [2] are forwarded from the source to the destination in a hop-by-hop store-and-forward manner.

The existing DTN routing protocols could be divided into two categories: knowledge-based and probability-based routing. Knowledge-based DTN routing schemes assume that certain information about the networks such as prior knowledge of the link connectivity pattern, the geographic locations, or the node movement schedules have been discovered by the networks. Then, routing decisions are made using Dijkstra-like algorithms to decide when and how a message should be forwarded. ED, MED, EDLQ and EDAQ methods proposed in [17], MV routing proposed in [18], and the message ferry routing and control schemes studied in [19, 20, 21] all belong to this category.

Probability-based DTN routing doesn't rely on any prior knowledge of the networks. In general, it requires that two nodes estimate the probability of successfully delivering a bundle to the destination by taking the other node as the next hop when they meet. How to estimate that probability differs in different ways. FC in [17] is the simplest one; it always trusts the first available contact to any neighboring node and forwards the bundles. However, loops could be easily found in FC. Epidemic Routing [22] makes two nodes exchange all the unknown messages to each other once they encounter each other. Its basic idea is based on the fact of that if a message is randomly distributed in the networks for enough time then it will eventually arrive at the destination. PROPHET [23] extends the idea by calculating the so-called delivery predictability based on the previous neighbors a node encountered. A message is forwarded only when the next hop is predicted to have a better chance to meet the destination in the future. However, such prediction could be inaccurate. Spray-and-Wait [24] is a more general approach than Epidemic Routing. It first sprays a message to a certain number of nodes, and then waiting for one of the copies to be directly forwarded to the destination. In this way, Spray-and-

wait is able to reduce the large amount of overheads introduced by Epidemic Routing. Also, different message spraying strategies could be developed and utilized according to the network conditions. Compare to knowledge-based routing approaches, probability-based DTN routing methods often have non-optimal route selections. Additionally, they usually introduce redundancies into the networks because multicast messages have to be widely distributed to intermediate DTN nodes between the source and the destinations.

Multicast for DTNs is a considerably different and difficult problem compared to that of Internet and mobile ad hoc networks (MANET), because of the frequency of network partitions. There are several issues of applying traditional multicast approaches to DTNs as discussed in Section I. DTBR [25] is a multicast scheme proposed for DTNs to dynamically build up a multicast tree. DTBR assumes that a certain knowledge oracle contains the schedule or the statistical summary of link up/down information and is known by each DTN node. Based on this, each node computes a multicast tree to the receivers listed in the receiver list in the header of multicast messages. If one node is a branching node, it then assigns a subset of the receiver list to each downstream next hop according to the calculated tree. DTBR is effective in terms of performing group communications. But it misses many opportunities to forward more bundles to the receivers as discussed in Section III.

The idea of our proposed approach, OS-Multicast, is similar to DTBR. It builds up a dynamic multicast tree hop-by-hop and tries to take advantage of any opportunity to forward a multicast bundle to all the receivers. However, OS-Multicast doesn't rely on any global knowledge of the networks, such as node positions or link up/down schedule. We only assume that the underlying network layer is able to record historical routing information and report the current availability of outgoing links to the upper-layer DTN multicast agent. Simulations show that OS-Multicast can deliver more messages to the multicast receivers with smaller average delays. Details of OS-Multicast are introduced in Section IV.

III. DTN MULTICAST APPROACHES

In this section, we present our assumptions of DTN multicast and illustrate three possible approaches including, multiple-unicast-based, static-tree-based and dynamic-tree-based multicast.

A. Network Model

We view DTNs as an overlay built upon certain underlying networks, such as mobile ad hoc networks. Only those nodes that implement DTN functionality to send and receive bundles are DTN nodes, while the others are denoted as normal nodes. The overlay, called the DTN layer in the rest of this report, consists of all the DTN nodes. It is also named as the bundle layer in [16].

In the DTN layer, two nodes are neighbors if at least one

end-to-end path between them exists in the underlying network. A DTN link may consist of several such paths in multiple hops, as depicted in Fig.1. The status of a DTN link is down if and only if all those possible paths are unavailable. Similarly, it is up if one of them is connected or reconnected or a new available path is discovered. The topology of the DTN layer dynamically changes due to the frequent link up/down variations. Clearly, an available DTN link represents an opportunity to forward bundles from one DTN node to another.

When there is no available outgoing DTN link, a DTN node holds bundles in its local buffer as required by the store-and-forward virtual message switching nature of DTN routing schemes. We assume that each DTN node has finite-size buffers for bundles it generates and receives. Those buffered bundles are sorted by the time when they are stored. Buffer management policy is needed to control how and when a bundle is deleted because the buffer size is limited. Obviously, different management strategies will affect the performance of DTN multicasting approaches.

We assume that the underlying networks are capable of unicasting a bundle from one DTN node to one of its DTN neighbors when there is an available end-to-end path between them. That traffic is viewed as a one-hop DTN transmission in the DTN layer, but it may traverse multiple hops in the underlying networks. We also assume that the underlying networks are able to cache historical routing information and report the current availability of outgoing links. The historical routing information is periodically updated by the underlying routing scheme.

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 an *endpoint ID* that allows late binding from the DTN layer to its underlying network address. The address translation between a DTN endpoint ID and its underlying network address is done by the DTN routing agent. A multicast source uses a group endpoint ID (or an explicit list of the receivers) as the destination address for multicast bundles.

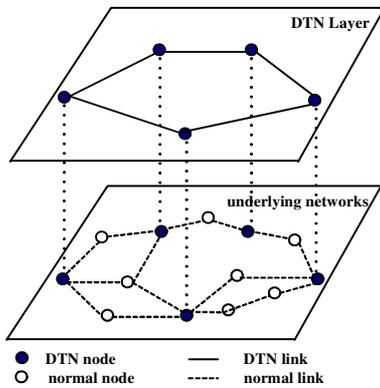


Figure 1. A simple example of DTN networks.

B. DTN Multicast Approaches

There are three possible approaches to support multicast service in disruption tolerant networks. All of them are performed only in the DTN layer. Comparing their performances is one of the major goals of this paper.

1) Multiple-unicast-based Multicast (U-Multicast)

The simplest way to do a one-to-many data communication is to conduct it by multiple one-to-one source-to-destination unicast data transfers. Any DTN unicasting method mentioned in Section II could be applied to perform this task. In this report, U-Multicast works in the following way: *i)* for each destination, the multicast source queries the underlying network to collect all the discovered historical end-to-end paths to that destination, finds the one with the smallest cost (e.g., the shortest path), and checks if the outgoing DTN link from the source in the smallest cost path is currently available; *ii)* if that link is up, a copy of the multicast bundle is then forwarded to the next hop; *iii)* otherwise, that copy is held in the buffer, waiting to be retransmitted once the link becomes available or a new data forwarding opportunity is discovered (e.g., a new shortest path is found); and *iv)* every DTN node that receives the bundle will work in the same way as the source. Clearly, if there are n receivers registered in a multicast group, then n copies of the bundle would be sent by the source and some intermediate DTN nodes may forward the same copy more than once.

2) Static-tree-based Multicast (ST-Multicast)

A spanning tree is a typical structure which is widely used by many multicast protocols. The usage of a multicast tree has been proven to be capable of reducing the redundant traffic [28]. In ST-Multicast, a tree is constructed and maintained at the source. When a multicast session starts, the source queries the underlying networks to find out the historical end-to-end paths to all the group members. It then builds up a smallest cost tree based on this information and keeps sending bundles along the tree. By definition, the topology of the (static-tree-based) multicast tree never changes in the DTN overlay during the multicast session. In ST-Multicast, a bundle is only duplicated at every branching node based on how many downstream neighbors it has. Intermediate nodes will just use the same tree to forward the bundles. When a branch is down, the upstream node will hold the bundle in its local buffer and retransmit it once that branch is up again. Obviously, the drawback of ST-Multicast is that it is not adaptive to dynamic topology variations in DTNs, especially when nodes may move. For example, in a disruption tolerant network consisting of highly mobile vehicles a static multicast tree will quickly become ineffective.

3) Dynamic-tree-based Multicast (DT-Multicast)

Unlike the ST-Multicast approach, DT-Multicast tries to dynamically adjust the multicast tree according to the current situation in the networks. In DT-Multicast, each bundle has a unique multicast tree and the tree may change hop-by-hop to adapt to the link up/down variations. For each node that has a multicast bundle (no matter whether the bundle is generated

by itself or received from others), it queries the underlying network to collect the discovered end-to-end paths to all the multicast receivers. Then it computes the smallest cost tree based on its current local view of the DTN layer and forwards the bundle using its own discovered multicast tree. In this way, if there is a newly available path to a destination which is not discovered by the upstream nodes, this node may immediately take advantage of that fresh information and send the bundle out. Also, the bundle forwarding follows the virtual message switching scheme. OS-Multicast as proposed here is a typical DT-Multicast method.

Fig. 2 illustrates a simple example of these three DTN multicast approaches. Assume that the link availability of the DTN layer is like Fig. 2(a) when the system starts. Then U-Multicast will forward three copies of the multicast bundle to the intended receivers using three DTN unicasting paths. However, ST-Multicast will set up a tree to cover the receivers and reduce the redundant traffic as shown in Fig. 2(b). Fig. 2(c) shows the advantage of DT-Multicast. If link $2 \rightarrow 5$ suddenly goes down but link $3 \rightarrow 5$ becomes available, DT-Multicast will take advantage of this opportunity immediately to make node 3 forward the bundle to receiver 5, rather than waiting for link $2 \rightarrow 5$ to be up again in ST-Multicast. And it is very possible that link $2 \rightarrow 5$ won't be available any more.

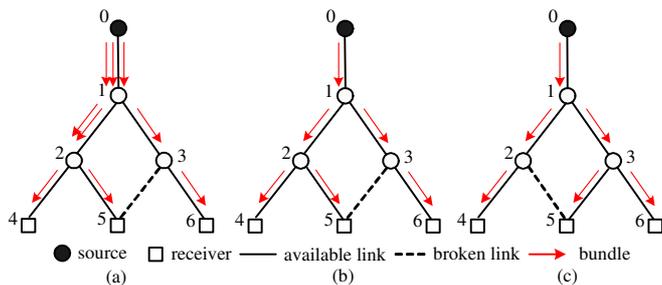


Figure 2. Different DTN multicast approaches. A multicast session has node 0 as the source and node 4,5,6 as intended receivers. (a) a snapshot of U-Multicast. (b) a snapshot of ST-Multicast. (c) a snapshot of DT-Multicast: when link $2 \rightarrow 5$ turns to be down but link $3 \rightarrow 5$ becomes available now, node 3 will take advantage of the currently available link immediately and forward the bundle. But in ST-Multicast, node 5 won't receive the bundle until link $2 \rightarrow 5$ is up again.

C. DTBR

DTBR is a DT-Multicast method proposed in [25]. The source first computes a multicast tree for each bundle based on the so-called knowledge oracle of the network. It then forwards the bundle along this tree. There is a receiver list associated with each copy of the bundle. It indicates for which receivers an intermediate node should be responsible. Initially, the list at the source contains all the intended receivers. If the source has more than one downstream node, it will put a new list that only consists of the receivers along that branch into the copy sent to each downstream next hop. Each node that receives a bundle will re-compute a multicast tree to reach those destinations in the receiver list. This

process is repeated until a copy of the bundle is delivered to a receiver.

[25] shows that DTBR is very effective in terms of conducting group communications. However, it misses some opportunities to forward more bundles to the receivers. *i)* DTBR splits the receiver list into subsets at each branching node in the dynamic tree. By this means, it reduces the redundant traffic but it also limits the potential of an intermediate node to utilize some currently available paths to reach receivers that are not included in the receiver list. Fig. 3(b) illustrates this issue by a simple example. Assume that when the multicast session starts, the source calculates a tree as shown in Fig. 3(a). The source then splits the receiver list and decides that node 1 is responsible for receiver 5, node 2 for receiver 6 and node 3 for receiver 7. Fig. 3 (b) shows the case in which link $2 \rightarrow 6$ is suddenly broken after node 2 receives the bundle. In this case, node 1 cannot take advantage of a currently available path $1 \rightarrow 4 \rightarrow 6$ to forward the bundle because receiver 6 is not in its receiver list. *ii)* DTBR will ignore some newly discovered paths. As depicted in Fig. 3(c), suppose that both $2 \rightarrow 6$ and $1 \rightarrow 4 \rightarrow 6$ are down but a new path $3 \rightarrow 6$ is up and discovered by node 3. In this case, neither can node 3 forward the bundle to receiver 6 because it is not node 3's responsibility to cover receiver 6. *iii)* Bundle forwarding in DTBR will automatically stop at each receiver. That also misses some opportunities to forward the bundle because one receiver may possibly meet another in the future, especially for DTNs consisting of mobile vehicles. Our proposed approach, OS-Multicast, tries to solve these issues and fully utilizes the bundle forwarding potentials.

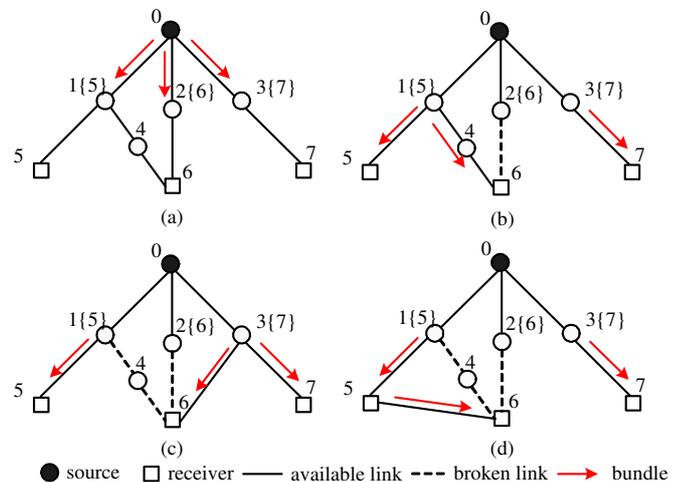


Figure 3. A simple example to show drawbacks of DTBR. (a) At time t_0 , the source (node 0) computes the multicast tree and decides that the receiver list for node 1 is {5}, for node 2 is {6}, and for node 3 is {7}. (b) Suppose at time $t_1 > t_0$, link $2 \rightarrow 6$ is down, but $1 \rightarrow 4 \rightarrow 6$ is available. (c) Suppose at time $t_1 > t_0$, both link $2 \rightarrow 6$ and $1 \rightarrow 4 \rightarrow 6$ are down but a newly available link $3 \rightarrow 6$ is discovered. (d) Suppose at time $t_1 > t_0$, a newly available path $5 \rightarrow 6$ is discovered by one receiver, node 5.

IV. THE OS-MULTICAST PROTOCOL

OS-Multicast (On-demand Situation-aware Multicast) is our

proposed DT-Multicast method. It is based on on-demand path discovery and situation awareness of the link availability to address the challenges of opportunistic connectivity in DTNs.

Situational awareness is achieved by making a DTN agent implemented in the DTN layer collaborate with the routing agent functioning in the underlying networks. The DTN agent will send a *situation_req* message to trigger its underlying routing agent to collect the current network conditions, such as the presently available outgoing links and the cached historical paths from the current DTN node to certain destinations. The underlying routing agent then answers with a *situation_rep* message with all the detected information back to the DTN agent to help make multicasting decisions. Both *situation_req* and *situation_rep* messages are system messages transmitted inside a DTN node. They are exchanged only when there is a bundle waiting to be transmitted/re-transmitted. The protocol of OS-Multicast is detailed as follows.

A. Bundle forwarding

When a multicast bundle is generated, the source queries its underlying routing agent to retrieve all the discovered paths destined to the receivers. By combining these paths together, a multicast mesh rooted at the source could be built to reach the receivers. We denote it as *static_mesh*. It records the historical routing information from the source to the other group members and is updated once the content in the routing cache of the underlying routing agent is refreshed. Then, a new mesh is generated by deleting those currently unavailable outgoing links from the *static_mesh* at the source. It is named *dynamic_mesh* because it may vary from bundle to bundle since the status of links could vary frequently. Based on the *dynamic_mesh*, the source is able to calculate a multicast tree to cover all the intended receivers. The source then forwards the bundle out to those available branches along the tree, associated with the full receiver list and the discovered *static_mesh*. Fig. 4 shows a simple multicast session with node 0 as the source and node 5, 6, 7 as the receivers. The *static_mesh* at node 0 is illustrated in Fig. 4(a) with those discovered paths. Fig. 4(b) depicts the *dynamic_mesh*, assuming link $0 \rightarrow 3$ is currently unavailable. Based on this, a multicast tree is built up as shown in Fig. 4(c) and the bundle is forwarded to node 1, which is the downstream node of node 0 in the tree.

Upon receiving a bundle from node X , a DTN node Y will also query its underlying routing agent to find the cached paths to the receivers. It combines the query results with the *static-mesh* in the bundle to be its own *static-mesh*. A *dynamic-mesh* is then constructed by deleting Y 's currently unavailable outgoing links and the link $X \rightarrow Y$ (to avoid the possible loop). After that, node Y re-computes a new multicast tree by taking itself as the root to all the receivers. This tree could be very different from the one generated by X because X and Y may have different views of the DTN layer. If Y is a branching node, it will duplicate the bundle and send a copy to each

downstream node. Node Y will also put its *static_mesh* into the bundles so that the knowledge could be propagated and used by other nodes that have poorer views of the DTN layer than Y . For example in Fig. 4(d), after receiving the bundle from node 0, node 1 put a new path $1 \rightarrow 4 \rightarrow 3 \rightarrow 6$ discovered by its own routing agent into the *static_mesh* and recalculates the multicast tree. From the tree, it finds that by forwarding the bundle to node 4, all the receivers may be covered.

B. Forwarding state maintainness

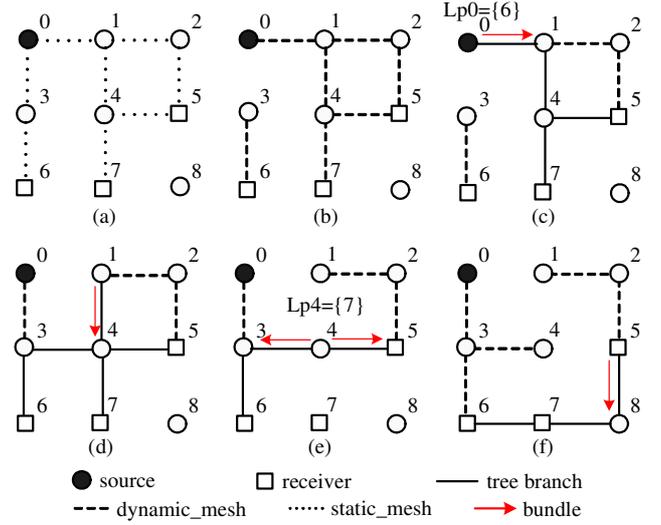


Figure 4. A simple example of OS-Multicast. (a) the *static_mesh* at the source, node 0, including all the discovered paths to the receivers: $0 \rightarrow 1 \rightarrow 2 \rightarrow 5$, $0 \rightarrow 1 \rightarrow 4 \rightarrow 5$, $0 \rightarrow 1 \rightarrow 4 \rightarrow 7$, and $0 \rightarrow 3 \rightarrow 6$. (b) the *dynamic_mesh* at node 0 by removing the currently unavailable outgoing link $0 \rightarrow 3$. (c) the multicast tree built up by node 0. Node 0 forwards the bundle to node 1 and its pending list is $\{6\}$. (d) the multicast tree at node 1, with adding a newly discovered path $1 \rightarrow 4 \rightarrow 3 \rightarrow 6$. Node 1 forwards the bundle to node 4. (e) the multicast tree at node 4, when node 4 detects that link $4 \rightarrow 7$ is currently down. It forwards a copy of the bundle to node 3 and another copy to node 5. (f) the multicast tree at node 5, with adding a newly discovered path $5 \rightarrow 8 \rightarrow 7 \rightarrow 6$. Node 5 forwards the bundle to node 8.

Each bundle has a unique forwarding state, including a *pending_list* (called L_p) and an *upstream_list* (called L_u). When a bundle arrives, a DTN node Y creates L_p for that bundle by copying the associated receiver list from the bundle. In OS-Multicast, L_p initially includes all the intended receivers for that bundle. After generating the multicast tree, Y knows which receivers could be covered using this tree. That means, from Y 's point of view of the DTN layer, these receivers should be reachable by this multicast tree. The covered receivers are then removed from L_p . If L_p is not empty, Y will put a copy of that bundle into its local buffer and wait for the future opportunities to cover those destinations in L_p . If L_p is empty, then the bundle doesn't need to be buffered since all the receivers have been covered from Y 's view of the network. For instance in Fig. 4(c), the multicast tree calculated at node 0 can only cover receiver 5 and 7. Thus, the source will put node 6 in the *pending_list* and retransmit the bundle once there

is an opportunity to reach receiver 6. Another example is shown in Fig. 4(e). After node 4 receives the bundle from node 1, it detects that link $4 \rightarrow 7$ is suddenly broken. Node 4 then has to put node 7 into the pending_list and keep a copy of that bundle for receiver 7 in its local storage. The upstream_list L_u contains the endpoint ID of DTN nodes a bundle has traversed. The purpose of L_u is to avoid the possible routing loops and reduce the redundant traffic. If a bundle has been processed by node X already, it won't be forwarded to X again.

C. Bundle storage

As mentioned in Section III, we assume that each DTN node only has finite local storage for both bundles it generates and receives. A buffered bundle may be released when *i*) its lifetime has expired; *ii*) its associated L_P is empty because there occurs a new opportunity to cover those receivers remaining in L_P ; or *iii*) the current buffer management policy decides to discard this bundle because of buffer overflow.

D. Bundle retransmission

Each DTN node periodically queries its underlying routing agent to check if there is any chance to forward the buffered bundles further. After each query, it generates a new dynamic_mesh to reflect the current network situations. It then recalculates a multicast tree to reach the uncovered receivers in L_P for each stored bundle. If one receiver is reachable by this new dynamic tree, it would be removed from L_P , and a copy of that bundle will be forwarded. For example in Fig. 4(c), if link $0 \rightarrow 3$ becomes available again, node 0 will forward the buffered bundle out to reach receiver 6 immediately. To reduce the overhead of 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 . Recall that once L_P is empty, the bundle will be removed from the local buffer.

E. Membership management

Membership management for DTN multicast is essential but difficult because join/leave messages may suffer large end-to-end delays caused by disruptions in the network. Several semantic models of DTN multicast membership have been studied in [25]. Our OS-Multicast conforms to one of them, the Temporal Membership semantic model, with an explicit receiver list known at the source.

When a DTN node intends to join a multicast group, it registers its planned membership period by explicitly flooding a *GROUP_JOIN* message into the DTN layer. For example, node i claims to be interested in the multicast service during the period $[t_{si}, t_{ei}]$, with the start-time t_{si} and the end-time t_{ei} . When the multicast source is informed by the *GROUP_JOIN* message, it puts the membership information into a *membership_list*, denoted as L_M . For each bundle the source generated or retransmitted at time t , it will check the validity of receivers in L_M . If the membership of a receiver has expired

($t > t_{ei}$) or was not activated ($t < t_{si}$), then that receiver will not be included in the receiver list in the bundle. An intended receiver with expired membership will be removed from L_M . The latest L_M is always forwarded with each multicast bundle. The intermediate nodes in the multicast tree manage L_M in the same way as the source.

When a receiver wants to leave the multicast session, it could leave silently. But a new *GROUP_JOIN* message is required when it wants to participate in the multicast service again in the future.

F. Differences between OS-multicast with DTBR

DTBR assumes that each node has complete knowledge or a summary of the link states in the networks. However, this is difficult to satisfy in most practical applications. In our design of OS-multicast, we require the underlying routing method record the historical routing information and detect the status of outgoing links. This situational awareness is achievable by putting extra effort into the routing implementations.

OS-multicast also gives each DTN node an equal chance to decide on the receiver list, rather than simply splitting the receiver list at each branching node in the multicast tree by DTBR. 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. For example in Fig. 4(f), node 4 holds a bundle for receiver 7 because link $4 \rightarrow 7$ is currently down. But when node 5 detects that there is a newly available path $5 \rightarrow 8 \rightarrow 7 \rightarrow 6$ and link $5 \rightarrow 8$ is currently up, it will at once forward the bundle to node 8 to get receiver 7 and 6 covered. The multicast bundle will arrive at node 7 as soon as either link $4 \rightarrow 7$ or link $8 \rightarrow 7$ becomes available.

The drawback of OS-Multicast is that it introduces more redundant traffic than DTBR. However one of the most important concerns of DTNs is how to overcome network disruptions to deliver more messages with fewer delays. Usually, there is enough link bandwidth between DTN nodes but little networking connectivity. OS-Multicast is designed to grab all possible chances to transmit bundles to destinations. To avoid broadcast storm, if the same bundle arrives at a DTN node more than once, it will be discarded.

V. PERFORMANCE COMPARISONS

To evaluate the performance of different multicast approaches in this study, we implemented U-multicast, ST-multicast, OS-multicast and DTBR in the *ns2* simulator [31]. Our performance metrics include: *i*) *message delivery ratio*, which is defined as the number of unique multicast bundles successfully delivered to all receivers over the total number of unique bundles which are expected to be received; *ii*) *message delivery 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*,

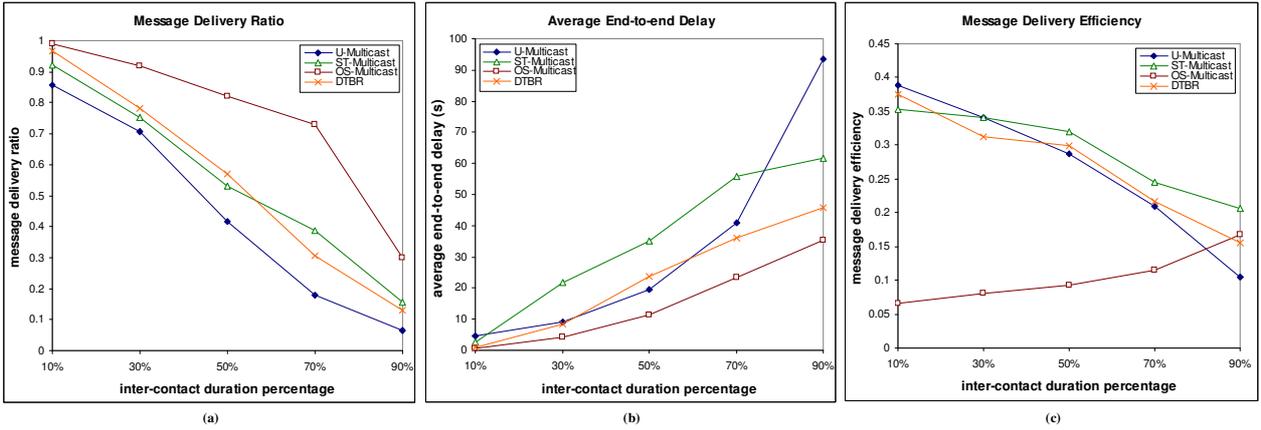


Figure 5. Simulation results of varying the total length of the inter-contact duration

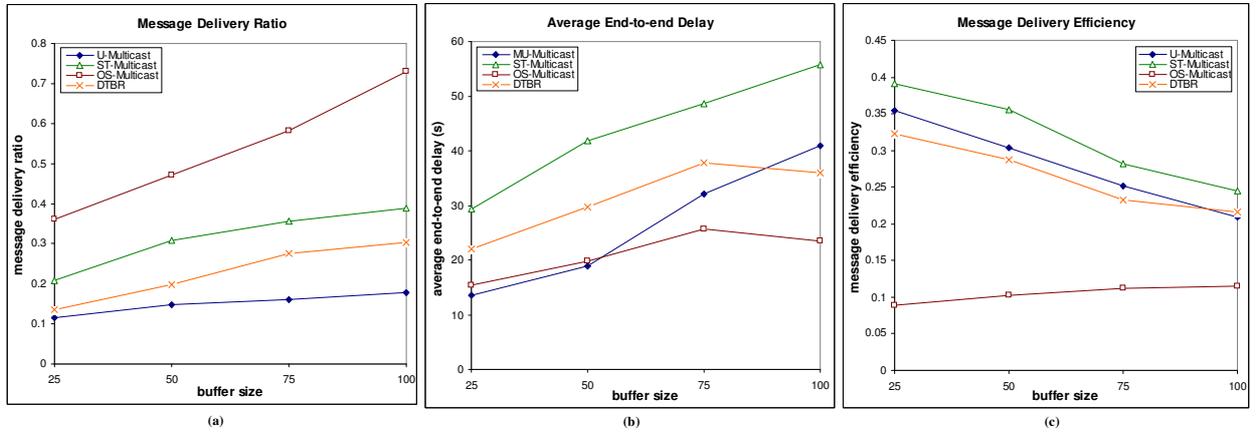


Figure 6. Simulation results of varying the local buffer size of each DTN node

which is the average of end-to-end bundle transmission delays for each algorithm.

All simulations are tested in a network with 25 nodes deployed in a 1000×1000 area. Those nodes could be configured as DTN nodes or normal nodes. But DTN multicast algorithms are only implemented in the DTN nodes. The DTN layer consists of all the DTN nodes. The MAC layer is IEEE 802.11 with radio transmission range of 250 meters. DSR [27] is chosen as the routing scheme for the underlying networks.

Simulations were conducted in different scenarios to study: *i)* the impact of the inter-contact duration; *ii)* the impact of the local storage size within DTN nodes; *iii)* the impact of different buffer management policies; *iv)* the impact of the scale of the DTN layer; *v)* the impact of group size and subgroup communications; and *vi)* performance evaluations by applying the real-world DTN traces, including PSN trace and DieselNet trace. Simulation results and discussions are as follows.

A. Varying the total inter-contact duration

In this study, 15 nodes are randomly selected to implement DTN functionalities. One multicast session is then applied in the DTN layer from the source to four other randomly chosen

receivers. The message sending rate is one 512 byte bundle per two seconds. Each DTN node can keep at most 100 bundles in its local buffer and try to retransmit those buffered bundles at every 5 seconds. The total length of the inter-contact duration of each link is varied from 10% to 90% of the overall simulation time.

Fig. 5(a) shows the message delivery ratio achieved by different DTN multicast approaches. We observe that *i)* fewer bundles could be delivered to the destinations when the inter-contact duration becomes larger, i.e., the network would be partitioned for longer time; and *ii)* OS-multicast delivers the most bundles because of its greedy nature of utilizing the currently available opportunistic links to push the data closer to the destinations.

Fig. 5(b) depicts the average end-to-end delays from the source to any receiver. When the lack of network connectivity becomes severe, both U-Multicast and ST-Multicast perform worse than the dynamic-tree-based multicast approaches. The reason is that ST-Multicast lacks the flexibility to dynamically maintain the multicast structure according to the network variations, and the shortest paths found in U-Multicast may have the longest inter-contact durations. Between the two DT-Multicast algorithms, OS-multicast has smaller delays than

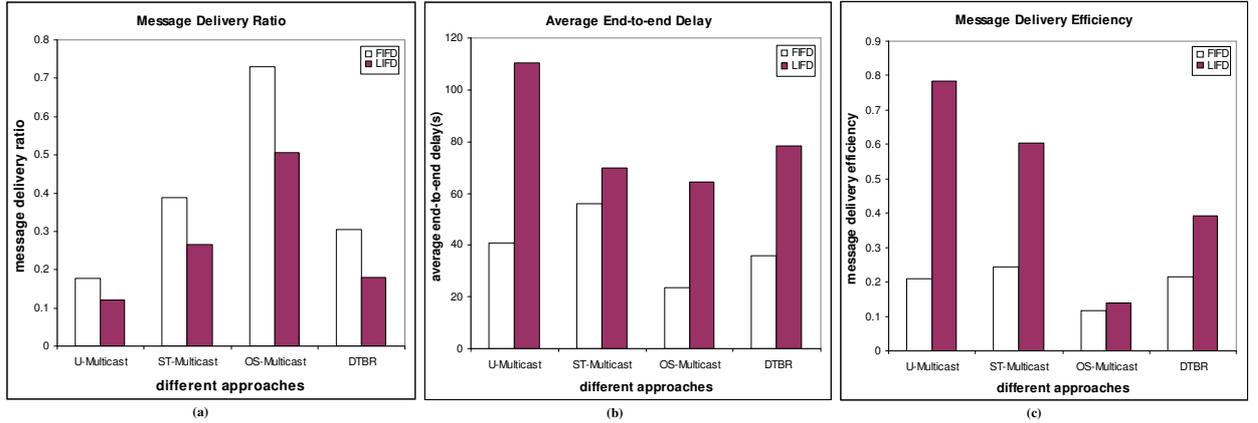


Figure 7. Simulation results of applying different buffer management policy

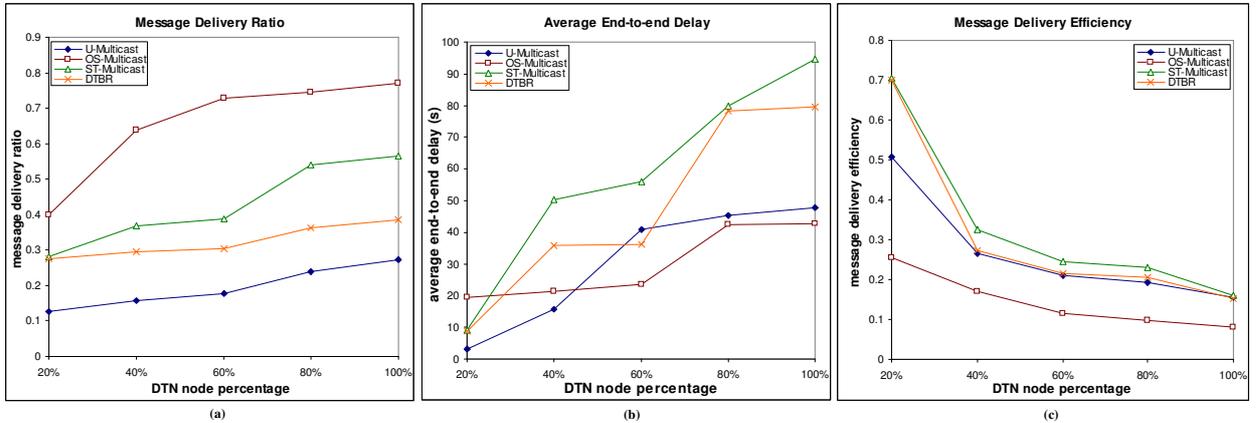


Figure 8. Simulation results of varying the number of nodes in the DTN layer

DTBR. As illustrated in Fig. 3, DTBR could miss some opportunities to forward bundles to the receivers that are not included in the receiver list decided by the upstream node based on its snapshot of the network conditions. However, OS-multicast always uses all the chances to forward bundles to the destinations.

The drawback of applying OS-Multicast is shown in Fig. 5(c). OS-multicast has the worst message delivery efficiency when inter-contact durations are small. It introduces a lot of redundant traffic because each intermediate node wants to fully take advantage of the currently available links to cover all the receivers. Thus, multiple copies of the same bundle may be delivered when the networks are connected most of time. However, as the network connectivity becomes very bad (e.g., at 80% time links may be down), the efficiency of OS-Multicast will benefit from its capability of delivering more bundles than the other three algorithms.

B. Varying the size of the local storage

DTN routing methods basically follow the store-and-forward virtual message switching scheme. Therefore, the size of local storage within each DTN node will affect the performance of different multicast approaches. If the size is

too small, some bundles have to be discarded due to buffer overflow. Those bundles then immediately lose the chance to be forwarded even when some opportunistic links would be available in the near future. However, if the buffer size is too large, it may be a waste of system resources. The optimal size of local storage is related to many network specifics such as the topology, the multicast group size, the traffic rate, and buffer costs, etc. But such a discussion of is out of the scope of this work.

Using the same scenario as Section A and fixing the total inter-contact duration to be 70% of the simulation time, we varied the buffer size of each DTN node from holding at most 25 bundles to 100 bundles. As shown in Fig. 6(a), basically with more local storage, more bundles could be delivered to destinations. Still, OS-Multicast achieves the largest message delivery ratio among all the multicast approaches. However, the average end-to-end delays illustrated in Fig. 6(b) also become larger when the size of local storage increases. This is because with more local storage, bundles could be buffered for longer time without being discarded until they get a chance to be forwarded. Thus, some of the delivered bundles may have been held in the network for a long time before reaching their destinations. And those bundles could be deleted due to buffer

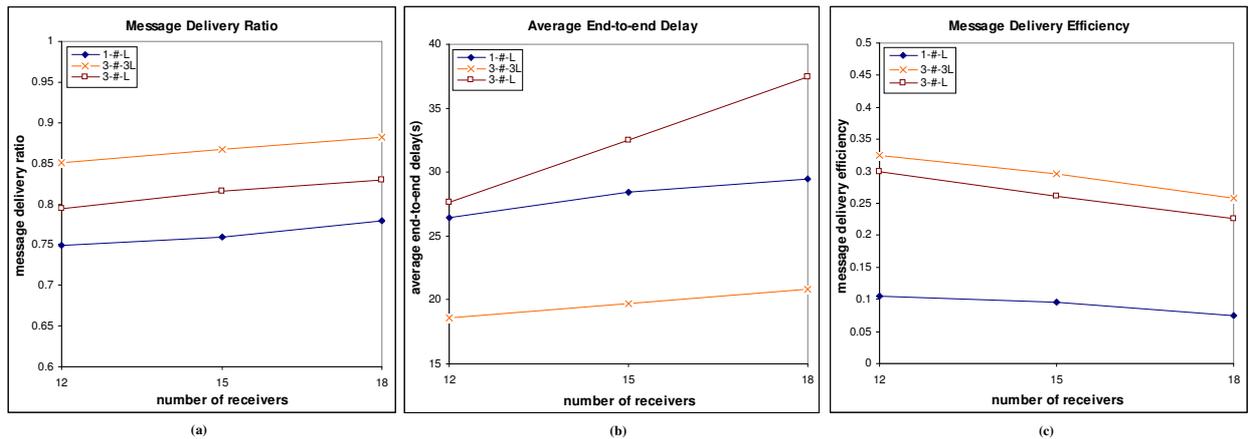


Figure 9. Simulation results of varying the group size

overflow with smaller size of local storage. Fig. 6(c) shows that the message delivery efficiency decreases when the buffer size increases, because more traffic is introduced into the networks, as a result of the retransmissions of more buffered bundles. The efficiency of OS-Multicast slightly improves since the increase of its message delivery ratio overcomes the drawback of incurring more redundant bundle forwarding traffic.

C. Applying different buffer management policy

A buffer management policy is to decide how and when a buffered bundle is removed from local storage. In this paper, we study two simple strategies: *i)* First In First Delete (FIFD) policy, which prefers to discard the eldest stored bundle once the local storage of one DTN node is full; and *ii)* Last In First Delete (LIFD) policy, which prefers to delete the latest incoming bundle while keeping the old ones. The former could be used for applications that favor the newest event reports from the multicast source while the latter is useful for applications in which the history data are more important.

Simulations are conducted in the same way as Section B with buffer size of 100. The results are depicted in Fig. 7. We observe that, for all the multicast approaches applying FIFD could deliver more bundles with smaller end-to-end delays in average than LIFD. By LIFD, old bundles will be held longer in the buffer so that they could have enough time to wait for the next chance to be forwarded. Thus, the newly generated/received bundles cannot be stored and forwarded until there is a space in the local storage. If we view the overall DTN layer as a big pipeline from the source to the receivers, LIFD will make the old bundles block the new ones from being transmitted for a longer period of time than FIFD. This is why LIFD has a smaller message delivery ratio and larger delays.

There are other more complicated buffer management policies. For example in [15], local storage of bundles is divided into two queues: one is for bundles generated by a DTN node itself and the other is for the received bundles.

Different priorities are applied to these two queues. Our observations tell that holding bundles too long in the local storages may hurt the performance of DTN multicast service.

D. Varying DTN node percentage

We are also interested in the impact of the scale of the DTN layer. As we discussed before, DTN functionalities are only implemented in those nodes that form the DTN layer. The other nodes in the network are just normal nodes which do not support any DTN service. Clearly, there is a trade-off between the performances of multicast approaches and the cost of deploying more DTN nodes (e.g., implementation cost, storage cost, etc.)

We study this issue via simulation by varying the number of DTN nodes from 20% to 100% of the total nodes in the network. As illustrated in Fig. 8(a), the message delivery ratios of all multicast approaches increase with more DTN nodes. When more nodes intend to support DTN functionality, there are more potential paths from the source to the receivers in the DTN layer. That means there could be more possibilities for bundles to find a chance of being forwarded closer to the destinations. And OS-Multicast performs much better than the other three approaches because it requires each DTN node to be aware of and to utilize these forwarding opportunities more aggressively than the other methods. However, with more DTN nodes, the lengths of paths in terms of the number of DTN hops could also increase. It indicates that bundles might be stored in the intermediate nodes more times. That's why the average end-to-end delays also become larger as shown in Fig. 8(b). Moreover, Fig. 8(c) shows that more bundle retransmissions would take place as the scale of the DTN layer increases. It brings down the message delivery efficiencies for all the approaches. One interesting observation of our simulations is that the message delivery ratio of OS-Multicast only becomes slightly better when the number of DTN nodes changes from 60% to 100% of the overall networks but the delay increases dramatically. It indicates that there is an optimal point of balancing the

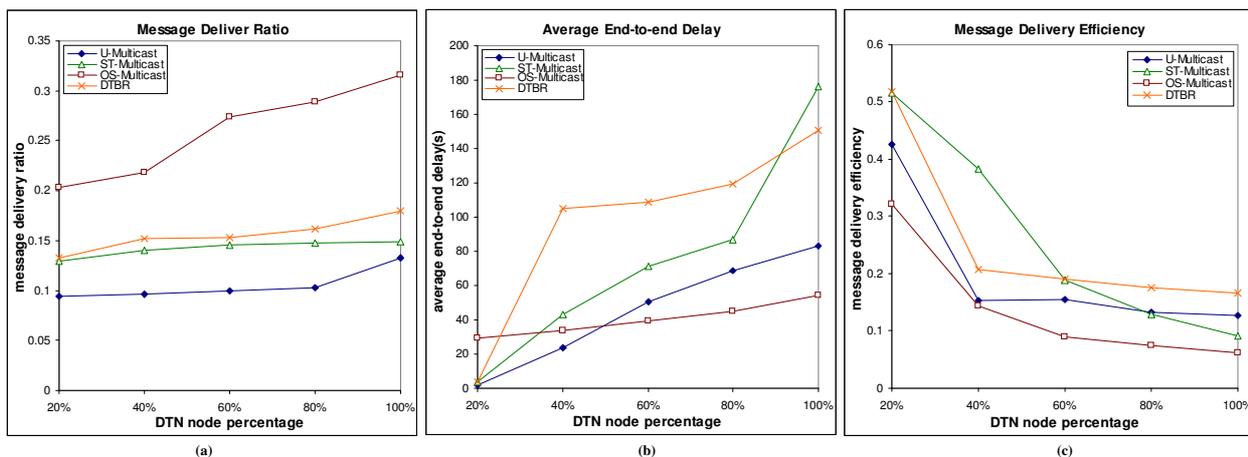


Figure 10. Simulation results of applying the PSN trace

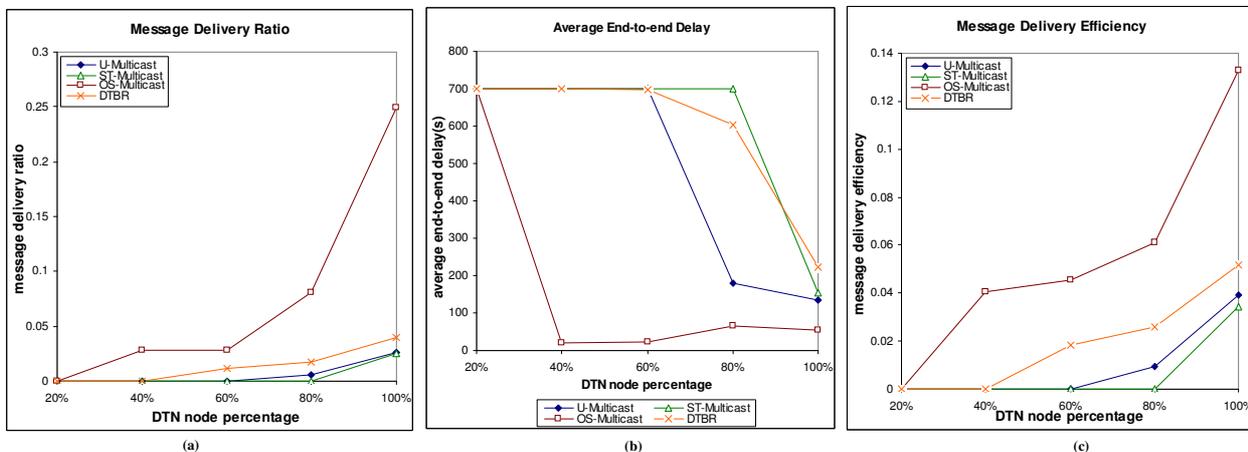


Figure 11. Simulation results of applying the DieselNet trace

performance of multicast method and the number of DTN nodes deployed in the networks.

E. Varying the group size

In this section, we want to study the impact of different group sizes to our proposed approach. We first simply increase the number of receivers of a multicast group from 12 to 18. We then divide a large DTN multicast group into three small sub-groups to transmit the same number of bundles in different ways. Assume that a multicast application has L bundles to be delivered to m receivers. This task could be done in the following ways: *i*) $1-m-L$, using OS-Multicast to build up a big multicast group consisting of the source and m receivers and dynamically maintaining the multicast structure among them in the DTN layer; *ii*) $3-m-3L$, to build up 3 small multicast sub-groups in which there are $m/3$ receivers and 1 source that is responsible for transmitting L bundles to those receivers in its group; and *iii*) $3-m-L$, to build up 3 small sub-groups and put the original source and 2 other randomly selected nodes to be the group leaders. In each group, there are $m/3$ receivers and 1 group leader. While transmitting

bundles to its own sub-group members, the source forwards L bundles to the other 2 group leaders by DTN unicast approach in the meantime. Once receiving a bundle from the source, the group leader then forwards that bundle to its sub-group members by OS-Multicast. The basic idea of $3-m-L$ is to conduct the multicast service in a 2-level hierarchy from group leaders to sub-group receivers. Scalable hierarchical multicast method has been proposed and studied in [30] for group communications in MANET

In our simulations, all the nodes are configured to be DTN nodes. Fig. 9 (a) shows that $3-m-L$ and $3-m-3L$ methods can deliver more bundles to the destinations than $1-m-L$. In $3-m-L$ and $3-m-3L$, there are two additional group leaders/sources compared to the method of $1-m-L$. By OS-Multicast, they are required to provide historical path information to the destinations in their *static_mesh* when multicast bundles are forwarded. The intermediate DTN node then has better knowledge about how the networks could be connected by combining the path information together. It helps them to find additional opportunistic links to forward the bundles to the

receivers. The result in Fig. 9(a) indicates that the message delivery ratio may benefit from dividing a large multicast group to several small sub-groups. Similar observations are reported in [29] for multicast service over the Internet. However, Fig. 9(b) shows that 3- m - L has the largest delay than the other methods. The additional delays are mainly caused by unicasting bundles from the original source to the group leaders of sub-groups. Without these additional delays, 3- m - $3L$ has smaller average end-to-end transmission delay than 1- m - L . Also, splitting a large multicast group to small sub-groups improves the message delivery efficiency because the intermediate node can construct a better dynamic multicast structure by merging the *static_mesh* from different group leaders. An extreme case of 3- m - L could be m - m - L , which turns out to be a variation of U-Multicast. Similar results of the average delays and efficiency can be found in Fig. 5.

F. Applying the real-world DTN traes

Clearly, different link up/down patterns affect the occurrences of opportunistic links in the DTN layer. In our previous simulations, the length of each link down period is uniformly randomly generated. That means the probability of having a short inter-contact duration is as same as that of having a very long one. However, practical DTN experiments such as the tests of the Pocket Switch Networks [14], and the DieselNet [15], have reported that the CDF of the inter-contact durations approximately follows a power-law distribution, by attaching networking devices to conference (Infocom 2005) attendees and buses around a city. It indicates that during the whole experiments, links are usually up for very short periods of time. In other words, the inter-contact durations could be longer in the practical disruption tolerant networks than our previous simulation scenarios. To further test the performance of all the multicast approaches, we apply these real-world traces into our simulations.

Fig. 10 and Fig. 11 illustrate the results of using the PSN trace and the DieselNet trace respectively. It is easy to tell that: *i*) with the real-world traces, message delivery ratios of all the multicast approaches are much smaller than the previous results shown in Fig.8. Obviously, it becomes more difficult to deliver bundles to the receivers when links could be repeatedly down for a very long period of time in the real world; *ii*) for each method, the average delay is larger than that in Fig. 8 because bundles have to be buffered in the intermediate nodes for longer time to wait for the reappearance of the broken links; and *iii*) OS-Multicast outperform the other DTN multicast approaches in terms of having the best message delivery ratio and the smallest average end-to-end delays. It demonstrates that OS-Multicast is capable of better utilizing the data forwarding opportunities that frequently vary in the networks than the other methods.

The results shown in Fig. 11 are more promising. In DieselNet, DTN devices called *Bricks* are mounted on buses moving around UMass Amherst. A DTN link is set up between two buses only when they encounter with each other

in their routes. Compared to PSN which attaches devices on people, the durations of links in DieselNet could be very short because buses move faster. Also, buses may move in different routes and are moving in a large area, other than conference attendees who usually sit in the same conference room. So the network connectivity of the DieselNet is worse than that of the PSN. Such network characteristics are also reflected in the simulation results we obtained. In Fig. 11(a), the ST-Multicast method cannot deliver any message to the destinations when DTN node percentage is less than 80%. Similarly, the U-Multicast scheme starts to deliver bundles to the receivers only when there are more than 60% DTN nodes in the network. However, the message delivery ratio of OS-Multicast quickly increases when more DTN nodes are deployed. For Fig 11(b), we assume that the message delivery deadline is set at 700 seconds so those messages that are not delivered by the deadline are considered failures. OS-Multicast is the first one that is able to find opportunities to forward bundles to the destinations. Also, OS-Multicast has the best message delivery efficiency as shown in Fig. 11(c). When network conditions are severe, its benefit of successfully transmitting more messages to the destinations outperforms the drawback of introducing redundant traffic into the network.

G. Discussion

In summary, our studies shows that: *i*) dynamic-tree-based DTN multicast approaches are better than U-Multicast and ST-Multicast methods in terms of message delivery ratio; *ii*) OS-Multicast is capable of better utilizing the opportunistic links than the other algorithms; *iii*) deploying more DTN nodes into the networks or having larger local storage within each DTN node can help improve the message delivery ratio of a multicast approach, but the average delays would also increase due to the potentially increased queuing delays; and *iv*) it is better to divide a large multicast group into several small sub-groups in DTNs to achieve better multicast performance.

Simulations also show that in most cases OS-Multicast has smaller message delivery efficiency than DTBR. It indicates that a hybrid approach of combining the benefits of OS-Multicast and DTBR could be developed. With this new method, a DTN node could switch between OS-Multicast and DTBR based on its current network conditions, such as the statistical information of the inter-contact durations of its outgoing links. When the inter-contact durations are larger than a certain threshold (e.g., in DieselNet), OS-Multicast is selected to take advantage of possible data forwarding opportunities. Otherwise, DTBR could be chosen to suppress the multicast redundancies while achieving decent message delivery ratio.

VI. CONCLUSIONS

In this article, we have discussed several multicast approaches that are applicable to disruption tolerant networks. We proposed an on-demand situation-aware multicast (OS-multicast) algorithm, which is able to dynamically adjust the

multicast structure in a hop-by-hop manner according to the current network conditions. Performance comparisons among OS-Multicast, DTBR, ST-multicast, and U-multicast are then conducted by simulation including tests using real-world DTN traces obtained from practical experiments. Simulation results show that OS-multicast is capable of achieving better message delivery ratio than existing approaches with smaller average delays. For example, as shown in Fig. 5, when the inter-contact duration percentage is 70%, i.e., links are statistically unavailable in 70% of the simulation period, OS-multicast achieves a bundle delivery ratio of about 80% but the peer protocols only achieve about or much less than 40%. When the lack of network connectivity becomes severe due to high link unavailability, OS-multicast also has better efficiency.

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] M. Chuah, L. Cheng, B. Davison, "Enhanced disruption and fault tolerant network architecture for bundle delivery (EDIFY)", to appear at *Globecom*, 2005
- [2] K. Fall, "A delay-tolerant network architecture for challenged Internets", *Proceedings of SIGCOMM'03*, August 2003
- [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] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] J. Moy, "Multicast extensions to OSPF", IETF RFC 1584, 1994.
- [10] D. Waitzman, C. Partridge and S. Deering, "Distance vector multicast routing protocol (DVMRP)", *IETF RFC 1075*, 1988.
- [11] 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.
- [12] 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.
- [13] Yong Wang, Sushant Jain†, Margaret Martonosi, Kevin Fall, "ErasureCoding Based Routing for Opportunistic Networks", in the *Proceeding of Sigcomm Workshop in DTN*, August 2005.
- [14] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot, "Pocket Switched Networks and Human Mobility in Conference Environments", in the *Proceeding of Sigcomm Workshop in DTN*, August 2005.
- [15] John Burgess, Brian Gallagher, David Jensen, B.N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking", in the *Proceedings of IEEE Infocom 2006*, April 2006
- [16] V. Cerf et. al., "Delay Tolerant Network Architecture", draft-irtf-dtnrg-arch-04.txt, December 2005
- [17] S. Jain, K. Fall, R. Patra, "Routing in a delay tolerant networking", in the *Proceedings of SIGCOMM'04*, Aug./Sep. 2004.
- [18] B. Burns, O. Brock, and B.N. Levine, "MV routing and capacity building in disruption tolerant networks", in the *Proceedings of IEEE INFOCOM'05*, pages 398-408, March 2005.
- [19] W. Zhao and M. Ammar, "Message Ferrying: Proactive Routing In Highly-Partitioned Wireless Ad Hoc Networks", in the *Proceedings of IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.
- [20] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad hoc Networks", in the *Proceedings of ACM Mobihoc*, May 2004.
- [21] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network", In the *Proceedings of IEEE INFOCOM'05*, 2005.
- [22] Amin Vahdat and David Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks", Duke Technical Report CS-2000-06, July 2000.
- [23] Anders Lindgreny, Avri Doria, and Olov Schelén, "Probabilistic Routing in Intermittently Connected Networks", in the *Proceedings of the First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, 2004.
- [24] Thrasyloulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks", in the *Proceeding of Sigcomm Workshop in DTN*, August 2005.
- [25] 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.
- [26] S. Symington, R. Durst, and K. Scott, "Non-Custodial (Best-Effort) Multicasting Support in DTN", draft-irtf-dtnrg-bundle-multicast-noncustodial-00, 2006
- [27] 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.
- [28] Randeep Bhatia, Li Li, "Characterizing Achievable Multicast Rates in Multi-Hop Wireless Networks", in the *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networks (MobiHoc'05)*, May, 2005.
- [29] S.K. Kasera, G. Hjalmtusson, D. F. Towsley, and J.F. Kurose, "Scalable reliable multicast using multiple multicast channels", *IEEE/ACM Transactions on Networking*, Vol. 8, No. 3, June 2000, pp. 294 -310.
- [30] C. Gui, P. Mohapatra. "Scalable Multicasting in Mobile Adhoc Networks", in the *Proceedings of IEEE Infocom 2004*, March 2004.
- [31] Network Simulator 2, ns2. <http://www.isi.edu/nsnam/ns/>