Computer Communications xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

An encounter-based multicast scheme for disruption tolerant networks

Yong Xi, Mooi Choo Chuah [*]

Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015, USA

ARTICLE INFO

Available online xxxx

Keywords:
Multicast
Disruption tolerant networking
Analysis
Delivery

ABSTRACT

Some ad hoc network scenarios are characterized by frequent partitions and intermittent connectivity. Hence, existing ad hoc routing schemes that assume that an end-to-end path exists from a source to a destination do not work in such challenging environment. A store-and-forward network architecture known as the disruption tolerant network (DTN) has been designed for such challenging network environments. Several unicast and multicast routing schemes have been designed for DTNs. However, the existing multicast routing schemes assume a route discovery process that is similar to the existing ad hoc network routing approach. Thus, in this paper, we design an encounter-based multicast routing (EBMR) scheme for DTNs which uses fewer hops for message delivery. We first describe how the EBMR scheme works and then present an analytical framework to estimate the delivery performance of the EBMR scheme. Next, we present some comparisons of the analytical and simulation results to show that our analytical framework provides delivery performance estimates that match closely the observed simulation results. Last but not least, we present simulation results to study the delivery performance of EBMR in different scenarios, e.g. different mobility models, different multicast group size, different number of multicast groups and different node speed. We also compare the performance of the EBMR scheme with other DTN multicast strategies. Our simulation results indicate that the EBMR scheme can achieve higher delivery ratio while maintaining high data transmission efficiency compared to other multicast strategies.

1. Introduction

With the advancement of technology, we can find many wireless devices, e.g. sensors, PDAs, laptops etc. Such devices can form wireless ad hoc networks dynamically without any infrastructure support. Much work has been done in the past to design unicast routing schemes for ad hoc networks [1,2]. However, the unicast routing schemes often assume that an end-to-end path exists between a source/destination pair and hence are not suitable for challenging network environments where the nodes experience intermittent connectivity and frequent partitions. In addition, much design has also been done for delivering multicast traffic in ad hoc networks, e.g. [3,4]. Again, such multicast routing schemes often assume a multicast tree can be maintained for delivering multicast traffic. It is difficult to maintain a multicast delivery tree in challenging network environments with frequent partitions and intermittent connectivity among the nodes.

Recently, a new network architecture [5] called the disruption tolerant network (DTN) has been proposed to allow partitioned nodes or clusters of nodes to communicate with one another. Recent research interests in this area include network architecture design [5,6] and different routing algorithms for DTNs [7–12]. Most of the routing schemes designed are delivering unicast traffic. However, many potential DTN applications require efficient network support for multicasting. For example, in a battlefield, soldiers in a platoon need to share information about their surrounding environment among one another. In a disaster rescue operation, it is critical to share information about victims and potential hazards among the rescue workers. Although group communication can be implemented by sending a separate copy of data via unicast to each user, this approach is not efficient and can potentially consume much power in small mobile devices that the soldiers or rescue workers carry. Thus, efficient multicast services are essential to support such applications.

Several DTN multicast routing schemes have been designed for DTNs, e.g. [13,14]. These DTN multicast routing schemes rely on a route discovery process that is similar to traditional ad hoc routing schemes and hence may not be able to perform well when the network becomes very sparse. In [15], the authors overcome this problem by allowing nodes in very sparse environment to use high power transmission to complete the route discovery process and use message ferrying to deliver data packets. They show that such an adaptive scheme can significantly improve delivery performance. However, this approach assumes that mobility of the nodes can be controlled which may not be

[*] Corresponding author. Tel.: +1 610 758 4061; fax: +1 610 758 4096.
E-mail addresses: chuah@cse.lehigh.edu, yox205@cse.lehigh.edu (M.C. Chuah).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.09.031

feasible in some network scenarios. Thus, we are interested in exploring a multicast delivery approach that does not require the nodes to have controlled mobility. Specifically, we explore an encounter-based multicast routing scheme that uses delivery predictability metric. Our contributions in this paper are: (i) we design an encounter-based multicast routing scheme that provides high delivery performance (high delivery ratio and data efficiency), (ii) we provide an approximate analysis for the average number of hops taken and the average per-hop delay for our scheme, (iii) we provide an extensive evaluation of our scheme in various scenarios, e.g. with different number of groups, mobility models etc and compare our scheme with other schemes.

The remainder of this paper is organized as follows. We provide a brief review of related work in Section 2. In Section 3, we describe several multicast strategies we consider in this work including the new encounter-based multicast routing scheme that we design for DTNs. In Section 4, we present an analytical framework for estimating the delivery performance of the EBMR scheme. We present analytical and simulation results to show that our analytical framework is useful in estimating the delivery performance in sparsely connected ad hoc network scenarios of interests to us. In Section 5, we describe our simulation setup and present our simulation results for different scenarios, e.g. different mobility models, different group size, different node speed, etc. We conclude in Section 6 with some discussions on future work.

2. Related work

2.1. Routing in intermittently connected networks

Several routing schemes have been proposed for DTNs [7–11]. These different schemes can be grouped into three categories. The first category [7] uses special nodes called ferries to deliver messages between partitioned networks. Ferry routes have significant effect on the data delivery performance, hence they need to be designed efficiently. The second category [9,10] uses multihop routing approach where contact history information is used to determine the next hop node to pass a message. For example, in [10], a probabilistic metric called delivery predictability is used to determine if a node needs to pass any stored messages to a new contact that it comes across. More discussions on this scheme will be provided in Section 3. The third category [11,12] uses a two-hop routing approach where the intermediate nodes that receive messages from any source have to store the messages until they can deliver the messages when they come into contact with the destinations of the messages. Sometimes, erasure-coding is used to encode and divide the message into multiple blocks and these different blocks are sent to different relays to increase the chances of a destination receiving a particular message since the destination only needs to receive a certain fraction of the encoded blocks to reconstruct the original message.

2.2. Multicast routing schemes for DTNs

Several multicast routing schemes have been designed for DTNs, namely (a) DTBR [13], (b) OS-multicast [14], and (c) context-aware multicast routing (CAMR) [15]. DTBR is a tree-based multicasting algorithm. DTBR assumes that each source node of the multicast group has complete knowledge or a summary of the link states in the network. During the lifetime of a multicast session, DTBR requires an upstream node to assign the receiver list for its downstream nodes based on its knowledge of the current network topology. The downstream nodes are allowed to forward bundles only to the receivers in the list. The custody transfer feature is enabled for those DTN nodes along the multicast tree. However, since the network topology changes frequently, it is not easy to maintain the multicast delivery tree. In addition, the receiver list cannot be adjusted by intermediate nodes once it is decided by upstream nodes, which means newly discovered delivery opportunities cannot be used by intermediate nodes.

Due to the limitations of DTBR, OS-multicast [14] was proposed. OS-multicast relies on a DSR-like routing to build a knowledge base of the link state and network topology. Unlike DTBR, OS-multicast let each intermediate node maintain a tree rooted at itself to all the receivers and adjust the receiver list according to local knowledge of the network topology. Via simulations, the authors [14] show that OS-multicast achieves good performance when the probability of the link unavailability is high. However, all simulations are based on a network of 25 nodes deployed in a $1000 \times 1000\ \text{m}^2$ area, which is still quite well-connected. The authors in [15] show that the performance of OS-multicast degrades when the network becomes sparser. Moreover, OS-multicast relies on a DSR-like route discovery process to build a knowledge base of the current network topology. Such a process will not work in a very sparse network environment.

In [15], the authors propose a CAMR scheme where nodes are allowed to use high power transmissions when the locally observed node density drops below a certain threshold. Each node maintains a 2-hop neighborhood information, and hence can deliver traffic without invoking a route discovery process if all receivers are within its two-hop neighborhood. In addition, the nodes are allowed to act as message ferries when they discover they are in a very sparse neighborhood. The combined high-power route discovery process and message ferrying features allow CAMR to achieve much higher multicast delivery ratio than DTBR and OS-multicast schemes. However, CAMR still relies on a route discovery process that is similar to the traditional ad hoc routing approach and also relies on the ability to control node movement. Thus, we are motivated to design a scheme that does not rely on the traditional route discovery process but purely based on node encounters. We describe our encounter-based multicast routing scheme in the next section.

3. DTN multicast strategies

In this section, we describe a few of the DTN multicast strategies that are considered in this paper as shown in Fig. 1 below:

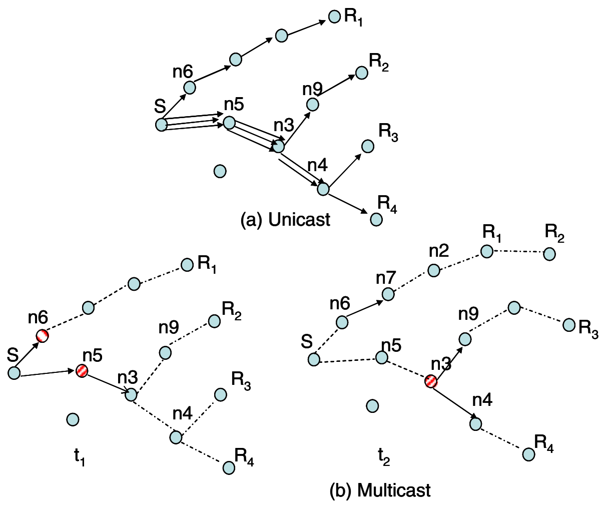


Fig. 1. Unicast and early decision-based multicast strategies.

3.1. Unicast-based multicast strategy

For unicast, the source duplicates one message per receiver and delivers it using the shortest route to each receiver. The diagram indicates that an intermediate node may receive multiple copies of the same message. Any DTN unicast routing scheme discussed in Section 2 can be used to delivery such unicast messages. In this work, we use the original Prophet scheme as the DTN routing scheme for the unicast approach.

3.2. Early decision-based multicast strategy

In this strategy, whenever a node, n_1, with a multicast packet to deliver encounters another node, n_2, it will check to see if that node has a higher delivery predictability value than itself to any receivers of that multicast packet. If the value from n_2 is higher, then a duplicate copy of the multicast packet will be sent to n_2. n_1 will keep a copy of that multicast packet until it has found a next-hop node for all the receivers of that packet. Thus, multicast messages are duplicated at every branching node based on how many downstream neighbors the node has in the multicast tree. For example in Fig. 1(b), at time t_1, S encounters two nodes, n_5 and n_6, simultaneously. Based on the delivery predictability values S hears from n_5 and n_6, S finds that n_5 has higher delivery predictability values to R_2, R_3, and R_4 than itself. Similarly, S finds that n_6 has higher delivery predictability value to R_1 than itself. Thus, S duplicates one multicast packet into two: one to be sent to n_6 with a header that includes the receiver R_1, the other which has a header that includes receivers R_2, R_3, and R_4 to be sent to n_5. n_5 encounters n_3 and finds n_3's delivery predictability to R_2, R_3, and R_4 to be higher than itself, so n_5 forwards its multicast copy to n_3. At time t_2, n_3 encounters nodes n_4 and n_9. n_4 has higher delivery predictability values to R_4 and n_9 has higher delivery predictability value to R_3. n_3 then duplicates two copies, one with a header that includes an identifier for R_3 to be sent to n_9 and another with a header that includes an identifier for R_4 to be sent to n_4. Since the multicast packet n_3 receives from n_5 has a header that also includes R_2, n_3 simply picks a node it encounters to send a multicast packet with a header that includes an identifier for R_2 when WT expires. Note that because early decision is made to split the receiver list, the multicast packet n_2 receives from n_7 only contains a header for R_1 and not R_2 even though n_2 may have higher delivery predictability value to R_2 at time t_2. Thus, with this strategy, some multicast packets may not reach the receivers due to too early a decision to split the receiver list in the header.

3.3. Encounter-based multicast routing (EBMR) scheme

Before we describe our EBMR scheme, we first give a summary on how the original Prophet [10] works. Prophet, a unicast DTN routing scheme, uses the history information of encounters and transitivity to select a next-hop node for message delivery. Specifically, Prophet establishes a probabilistic metric called delivery predictability at every node A for each known destination B. This metric indicates how likely it is that node A will be able to deliver a message to that destination. Each node broadcasts a beacon periodically. The beacon contains the delivery predictability values from this node to all other nodes. Such delivery predictability values are updated upon receiving beacons from other nodes. The delivery predictability value ages with time and also has a transitive property, i.e., a node A that encounters node B which encounters node C allows node A to update its delivery predictability to node C based on its (A's) delivery predictability to node B and node B's delivery predictability to node C. In Prophet, a node will forward a message to another node it encounters if that node has higher delivery predictability to the destination than itself. Such a scheme was shown to produce superior performance than epidemic routing [10]. The three equations used for updating the delivery predictability are as follow:

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) * \alpha, \tag{1a}$$

$$P(a,b) = P(a,b)_{old} \times \gamma^k, \tag{1b}$$

$$P(a,c) = P(a,c)_{old} + (1 - P(a,c)_{old}) * P(a,b) * P(b,c) * \beta, \tag{1c}$$

where $P(a,b)$ denotes the delivery predictability of reaching node b from node a and α is an initialization constant chosen from the range $[0,1]$. Eq. (1a) allows a node to update the metric whenever a node is encountered so that nodes that are often encountered have a high delivery predictability. If a pair of nodes does not encounter each other for a while, they are less likely to be good forwarders of messages to each other, thus the delivery repredictability values must age. The aging equation is shown in Eq. (1b) where γ is the aging constant and k is the number of time units that have elapsed since the last time the metric was aged. The delivery predictability also has a transitive property that is based on the observation that if node a frequently encounters node b, node b frequently encounters node c, then node c probably is a good node to forward messages destined to node a. Eq. (1c) shows how this transitivity affects the delivery predictability where β is a scaling constant that decides how large impact the transivity should have on the delivery predictability. In [10], α is set to 0.75, β is set to 0.25 and γ is set to 0.98. Each node maintains an N*N matrix (where N is the total number of nodes in the system) where each row i records the delivery predictability of node i to the other $(N-1)$ nodes (the diagonal entry (i,i) is not used). Every time a node's beacon timer expires, that node will use Eq. (1b) to update the delivery predictability values of those nodes that it has lost contacts with. If a node, n_1, hears another node (say n_2)'s beacon which contains the delivery predictability values from that node to other $(N-1)$ nodes, then n_1 uses those values it hears from n_2, Eqs. (1a) and (1c) to update its own delivery predictability values to other nodes.

Via simulation studies, we find that the Prophet scheme often uses large number of hops to deliver unicast messages. This means many transmissions are incurred in delivering a single message. With limited battery power resources, a more energy-efficient routing scheme needs to be designed, especially so when one needs to deliver multicast messages.

Next, we describe our EBMR scheme works. Our scheme uses the same equations as Prophet to compute the delivery predictability but includes new features that can delivery messages using fewer hops. Note that instead of using the same equations as Prophet to compute delivery predictability, we can use other approaches, e.g. [9] for delivery cost calculations in EBMR. In EBMR, each node will not pass any bundle to another node that it encounters unless that node has delivery predictability higher than a delivery threshold (P_{thresh}) or a wait timer (WT) expires. The wait timer prevents the messages from being held up for too long at the source or any intermediate node. When a node using the EBMR scheme receives a multicast bundle, it will pick as many nodes as needed with the highest delivery predictability (that exceeds P_{thresh}) to each of the multicast receivers. The node will cache the data if no such next-hop node is found until a wait timer WT expires. If WT expires, then the node will simply pick a node with the highest delivery predictability to a multicast receiver whose next-hop node has not yet been selected. These two enhancements improve the delivery performance. Every relay node (including the source node) can have the opportunity to select a next-hop node with delivery predictability higher than P_{thresh} within the WT time. This results in fewer numbers of hops being taken to deliver messages to all multicast receivers. Thus, the overall delivery ratio and data efficiency of EBMR should be better than the early decision-based multicast strategy.

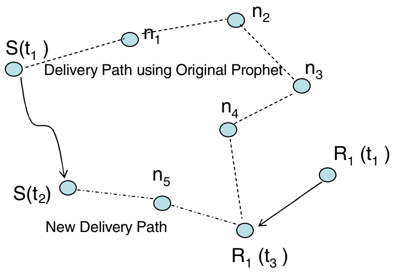


Fig. 2. Paths taken using original Prophet and our enhancement.

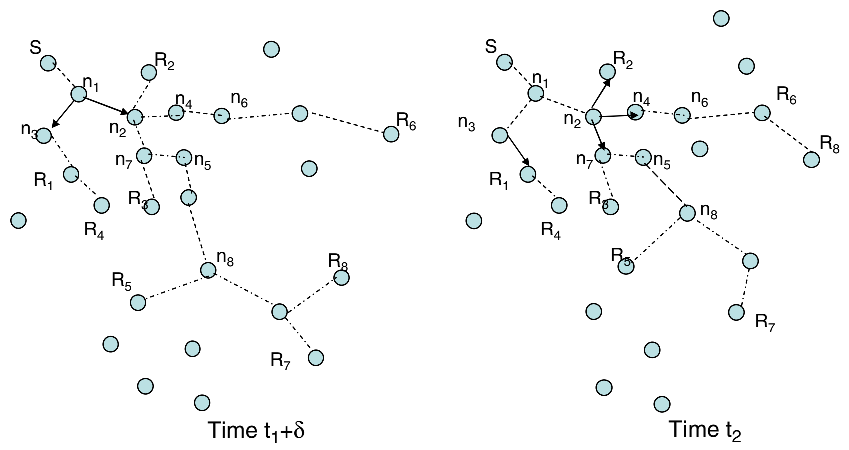
Our WT and P_{thresh} enhancements work for unicast scenario as well. In Fig. 2, we show how the inclusion of WT and P_{thresh} in our scheme results in a different delivery path for a unicast scenario. Using the original Prophet scheme, the message from *S* will take five hops to reach R_1. However, using our enhanced scheme, the source, *S*, will decide to hold onto the message initially because it has not encountered any node with delivery predictability that exceeds P_{thresh}. When *S* meets node n_5 with a delivery predictability that exceeds P_{thresh}, then it will choose n_5 as the next hop. Such decision allows the message to be delivered to R_1 in two hops. Of course, depending on the node movements, sometimes, it is not fruitful to hold onto the packets. However, our simulation studies reveal that our enhancement does produce better delivery ratio and average message delivery latency results.

Next, we describe how our scheme delivers multicast packets. In Fig. 3, we show a source *S* that sends a multicast packet to eight receivers, R_1 to R_8, respectively. In Fig. 3, *S* encounters n_1 at time t_1 so *S* sends n_1 a multicast packet with a header that includes the identifiers for all eight receivers. n_1 encounters nodes n_2 and n_3 at time $(t_1 + \delta)$. n_1 will look at the rows corresponding to n_2 and n_3 in the $N \times N$ delivery predictability matrix it maintains, and see which one has a higher delivery predictability value for a particular receiver. The predictability value to any receiver has to be higher than P_{thresh} (which is set to say 0.1) before that node can be considered as a suitable next-hop node. Using the table shown in Fig. 3, one can easily see that node n_2 should be used to reach receivers R_2, R_3, and R_5 while node n_3 should be used to reach receivers R_1 and R_4. Thus, n_1 will create two multicast packets: one with a header that includes the identifiers for R_2, R_3, and R_5 to be sent to n_2; the other with a header that includes identifiers for R_1 and R_4 to be sent to n_3. Node n_1 records the information that no suitable next hop node has been found for receivers R_6, R_7, R_8 yet at time $t_1 + \delta$. At time t_2, let us assume that the nodes have moved such that the delivery predictabilities for R_6, R_7 via n_2 have increased above P_{thresh}. Then, node n_1 will create yet another multicast packet with a header that includes the identifiers for R_6 and R_7. Note that at time t_2, node n_1 still has not found a suitable next hop node for R_8 with the delivery predictability that exceeds P_{thresh}. When WT expires, node n_1 will select a next-hop node with the highest delivery predictability to R_8 (even if this value does not exceed P_{thresh}) and forwards a copy of the multicast packet to that node. Node n_1 only removes this multicast packet from its buffers after it has successfully selected a next hop node for all receivers.

Note that if one uses the early decision-based multicast strategy described in Section 3.2, then, every intermediate node that receives a multicast packet will select the best next-hop node for all the receivers using the $N \times N$ matrices that it maintains, and will duplicate as many packets as needed to be sent to the selected next-hop nodes that will reach all receivers. In our Fig. 3 example, at time t_1, node n_1 will only generate two multicast packets: one with a header that includes identifiers for R_2, R_3, R_5, R_6, R_7, and R_8, and the other with a header that includes identifiers for R_1 and R_4. Since the delivery predictability for R_6, R_7 and R_8 from n_2 is low at time $t_1 + \delta$, such early decision may result in the duplicated multicast packets taking longer paths to reach these three receivers. Our simulation results show clearly that EBMR performs better than the early decision-based multicast strategy described in Section 3.2.

3.4. EBMR with replication

Replication forwarding has been shown [9] to improve DTN message delivery performance in unicast scenarios. Thus, an additional enhancement one can use with EBMR is to allow a source node to select *K* next-hop nodes for each multicast receiver. In Fig. 4, we show how EBMR with replication strategy works. Only the source is allowed to select *K* (*K* = 2 in Fig. 4) next-hop node for each multicast receiver. Let us assume that the source *S* wants to send a multicast packet to the four receivers R_1 to R_4. At time t_1, based on the $N \times N$ delivery predictability matrix *S* maintains, *S* selects nodes n_1 and n_2 as the next hop nodes for the first copy of a multicast packet. At time t_2, nodes n_1 and n_2 forwards the multicast packet they received to nodes n_4 and n_6, respectively. Meanwhile,



Nxt Hop	R_1	R_1	R_3	R_4	R_5	R_6	R_7	R_8
n_2	0.001	0.33	0.2	0.001	0.1	0.012	0.002	0.002
n_3	0.25	0.001	0.001	0.15	0	0.001	0	0

Nxt Hop	R_1	R_1	R_3	R_4	R_5	R_6	R_7	R_8
n_2	0.001	0.33	0.2	0.001	0.15	0.11	0.11	0.05
n_3	0.25	0.001	0.001	0.15	0	0.001	0	0

Fig. 3. Encounter based multicast routing (EBMR) scheme.

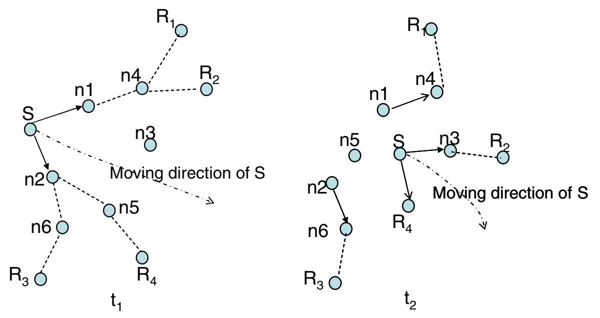


Fig. 4. EBMR with replication (MK = 2).

S has moved to a new location and determined that it can reach R_4 directly, and that n_3 can be used to reach R_2. As S has so far selected only one next hop node n_1 (n_2) for R_2 (R_4), respectively, therefore S can send the multicast packet directly to R_4 and send another copy to n_3 to be delivered to R_2 since $K = 2$. It is obvious from Fig. 4 that using replication with EBMR scheme allows us to have shorter delivery latency at the expense of more transmissions.

4. Analysis of EBMR

In this section, we present an analytical framework which estimates the delivery performance of EBMR when the nodes move according to the RWP model. This analytical framework allows us to investigate how the choices of the two tunable parameters in EBMR namely P_{thresh} and WT, affect the delivery performance. Since we are interested only in the sparsely connected ad hoc network scenarios, our analysis assumes that the node density is low, e.g with an average number of neighbors that is below 1.5. We first present an analysis for determining the average number of hops, L_m, taken to reach a receiver with different P_{thresh} and WT values. Then, we present an approximate analysis for the average one-hop delay, d_{hop}. Such analysis allows us to estimate the average end-to-end delay. Next, we present simulation results to show that the approximate analysis we have is close to the observed simulation results. Last but not least, we present simulation results that reveal the sensitivity of the delivery performance with different values of the two tunable parameters.

4.1. Analysis for L_m

Given P_{thresh}, WT, and other network parameters (like, $N \times N$, M, v_{ave}, R, etc.), we estimate the average number of hops, L_m, when EBMR is used. Table 1 summarizes the notations we use.

1. Determining p_i, the delivery probability towards the destination after one hop.

In EBMR (P_{thresh}, WT), a node holding a packet may have two delivery choices:

(a) Before WT expires, it may send the packet out when encountering a node with a delivery probability to the destination higher than P_{thresh}.
(b) When WT expires, it will simply pick up a node with the highest delivery probability to the destination.

Suppose during WT, a node will encounter n_e nodes. n_e can be calculated below [18]:

$$n_e = 2 * R * v_{ave} * WT * d, \quad (2)$$

where d is node density. Then, denote:

p_0: the probability that none of these n_{e-WT} encounters has a delivery probability that exceeds P_{thresh}. In other words, a node has to hold the packet until WT expires.
p_i: the average delivery probability towards destination after one hop.
p_e: the new delivery probability towards destination if a packet is relayed before WT expires.
p_d: the new delivery probability towards destination if the packet is relayed at the expiration of WT.

Therefore,

$$p_i = (1 - p_0) * p_e + p_0 * p_d. \quad (3)$$

2. Determining L_m.

Next, we use a similar argument as in [2] to derive the average hop L_m. We denote lm as the number of hops a packet takes to reach its destination, and let $P(lm > \text{n})$ be the probability that the packet has not reached the destination after n hops. We then have

$$P(lm > n) = \prod_{i=1}^{n}(1 - p_i). \quad (4)$$

Since each hop is independent of one another, we have

$$P(lm > n) = (1 - p)^n, \quad \text{where } p_i = p. \quad (5)$$

Eq. (5) indicates that lm is geometrically distributed and hence $E[lm] = L_m = 1/p$.

3. Derivation of p_0,p_e,p_d.

(a) Determining p_0.
p_0 can be derived as follows:

$$p_0 = \prod_{i=1}^{n_{e|WT}} p_{0|i}, \quad (6)$$

Table 1
Notation.

$N \times N$	Network area
M	Number of nodes
v_{ave}	Average speed of node movement
R	Transmission range
P_{thresh}	Delivery probability threshold
WT	Wait timer in EBMR
EM	Average node meeting time

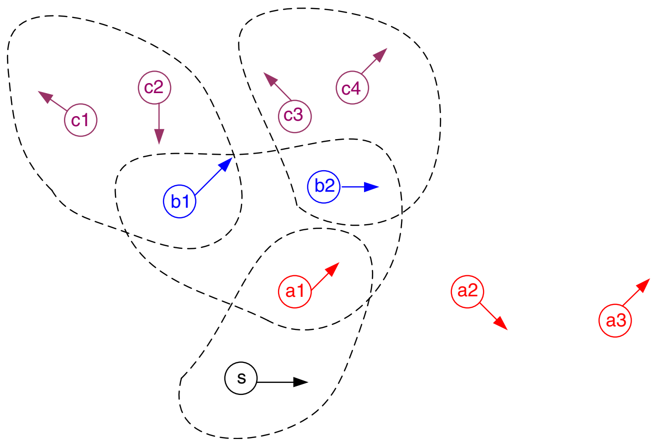


Fig. 5. Node encounters.

where

(i) p_{0-i} means the probability that the ith encounter does not have delivery probability higher than P_{thresh}. In other words, the packet will not be sent to the ith encounter.
(ii) n_{e-WT} is the number of encounters during WT.

To understand how to derive p_0, consider the picture in Fig. 5. Let t_a be the time taken for the delivery probability to a destination to drop from 1 to P_{thresh} when a node no longer can hear any nodes that can reach the destination. Let us assume that the source node encounters node a1 at time t_1, nodes b_1,b_2 were encountered by node a_1 during $[t_1 - t_a, t_1]$, nodes c_1,c_2 were encountered by node b_1 during $[t_2 - t_a, t_2]$ where $t_2 \in [t_1 - t_a, t_1]$ is the time node a_1 encounters node b_1, and nodes c_3, c_4 were encountered by node b_2 during $[t_3 - t_a, t_3]$ where $t_3 \in [t_1 \ - t_a, t_1]$ is the time node a_1 encounters node b_2. Then, given P_{thresh}, node S will not relay the packet to node a1 as long as none of these nodes b_1, b_2, c_1, c_2, c_3, and c_4 encounter any receiver during the indicated periods.

Therefore, we have

$$p_{0|i} = \frac{\binom{M-2-(i-1)}{1}}{\binom{M-1-(i-1)}{1}} \left(\frac{\binom{M-2}{n_{e|t_a+T_c}-1}}{\binom{M-1}{n_{e|t_a+T_c}-1}} \right)^{(n_{e|t_a+T_c})} \tag{7}$$

where i=1, ... , $n_{e|WT}$, t_a is the aging duration.

$$t_a = T_s * \frac{\log P_{thresh}}{\log \gamma}, \tag{8}$$

where γ is the decaying factor used in updating the delivery predictability, T_s is the beacon period, $n_{e|(t_a+T_c)}$ is number of nodes encountered during t_a+T_c where T_c is the contact duration. Eq. (8) can be obtained easily from Eq. 1c by noting that the delivery predictability is updated every T_s seconds and that the value of $P(a,b)$ is close to 1 when nodes a and b are in contact with each other for at least two beacons ($2T_s$ seconds) before they depart. Thus, $\log P_{thresh} = \log \gamma^k$ and $k = \log P_{thresh} / \log \gamma$. Since a beacon is transmitted every T_s seconds, and Eq. (1b) is updated every beacon period, therefore Eq. (8) holds.

(b) Determining p_e.

Recall that p_e is the new delivery probability of a packet towards destination after being delivered to a node with the probability higher than P_{thresh}.

To calculate p_e, we rewrite the aging function as follows:

$$y = G(x) = \gamma^x, \tag{9}$$

$$x = G^{-1}(y). \tag{10}$$

Eq. (9) assumes that the delivery predictability between two nodes immediately before disconnecting is close to 1 before it starts to age which holds most of the time when two nodes a,b are in contact for at least 2–3 beacon periods. Therefore,

$$p_e = E[G(x)] = \int_0^{G^{-1}(P_{thresh})} G(x)f(x)dx, \tag{11}$$

where $f(x)$ is the probability density function of x. Since x is uniformly distributed between $(0, G^{-1}(P_{thresh}))$, we have $f(x) = 1/G^{-1}(P_{thresh})$. Hence,

$$p_e = \frac{1}{G^{-1}(P_{thresh})} \int_0^{G^{-1}(P_{thresh})} G(x)dx. \tag{12}$$

(c) Determining p_d.

p_d is the new delivery probability of a packet towards destination after being delivered to a node at the expiration of WT (we assume in this case, the delivery probability is lower than P_{thresh}).

As before, we have

$$p_{d'} = E[G(x)] = \int_{x=G^{-1}(P_{thresh})}^{\infty} G(x)f(x)dx, \tag{13}$$

where $f(x)$ is probability density function of x, when $x \in (G^{-1}(P_{thresh}), EM/T_s)$ where EM is the expected meeting time derived in [19], and T_s is the beacon period. Therefore,

$$p_d = \frac{1}{EM/T_s - G^{-1}(P_{thresh})} \int_{G^{-1}(P_{thresh})}^{EM/T_s} G(x)dx. \tag{14}$$

From [19],

$$EM = \frac{1}{1.75p_m + 2(1-p_m)} \frac{N^2}{2R\overline{L}} (\overline{T} + \overline{T}_{stop}), \tag{15}$$

where $p_m = \overline{T}/(\overline{T} + \overline{T}_{stop})$, $\overline{T}$ is epoch duration, $\overline{L}$ is epoch length, $\overline{T}_{stop}$ is the pause time.

We did some simulation studies to compare the average number of hops value seen in the simulation with the derived L_m value. We use a network scenario with 40 nodes distributed over $3000 \times 3000\ \text{m}^2$. The nodes move according to the random waypoint model with speed randomly chosen between $(1,5)$ m/s. The transmission range used is 250 m. In the first set of experiment, we fix WT at 500 s and vary P_{thresh} from 0.2 to 0.9. In Fig. 6(a), we compare the analytical and simulation values for the expected hop count for this set of experiment. Our results show that the two values match closely. The results also show that when WT is set to 500 s, setting P_{thresh} to 0.5 already allows us to reach the smallest expected hop count value. In our second set of experiment, we fix P_{thresh} to 0.2 and vary WT. The analytical and simulation values for the second experiment are plotted in Fig. 6(b). Again, we see that the analytical and simulation values match closely.

4.2. Approximate analysis for d_{hop}

d_{hop} can be derived as follows:

$$d_{hop} = \sum_{i=1}^{n_e} q_i * Q(i) + \prod_{j=1}^{n_e} p_{0|j} * (WT + 0.5t_{nc}) \tag{16}$$

where q_i is the probability of a packet is delivered to the ith encounter during the WT $Q(i)$ is the waiting time to meet this ith encounter, and t_{nc} is the average node encounter time. Recall that p_{0-j} can be computed using Eq. (7).

$$q_i = (1 - p_{0|i}) \prod_{j=1}^{i-1} p_{0|j}, \quad \text{where } i = 1, \ldots, n_e. \tag{17}$$

We approximate $Q(i)$ as $E(w)$+$(i-1)E(y)$ where w is the residual node encounter time and y is the node encounter time. We assume that the node encounter time is exponentially distributed, and hence $E(w)$=$E(y)$=t_{nc} Our simulation results indicate that this is a relatively good approximation. Thus,

$$d_{hop} = \sum_{i=1}^{n_e} q_i * i * t_{nc} + \prod_{j=1}^{n_e} p_{0|j} * (WT + 0.5t_{nc}). \tag{18}$$

If we assume that the per hop delay is independent, then the average end to end delay, d_{e2e} will be $Lm * dhop$. Since we assume sparse networks where the average internode encounter time is long, the independent per hop delay assumption is justified.

In Fig. 7(a) and (b), we plot the analytical results we obtained from Eq. (18) as we vary P_{thresh} or WT, respectively. Next, we

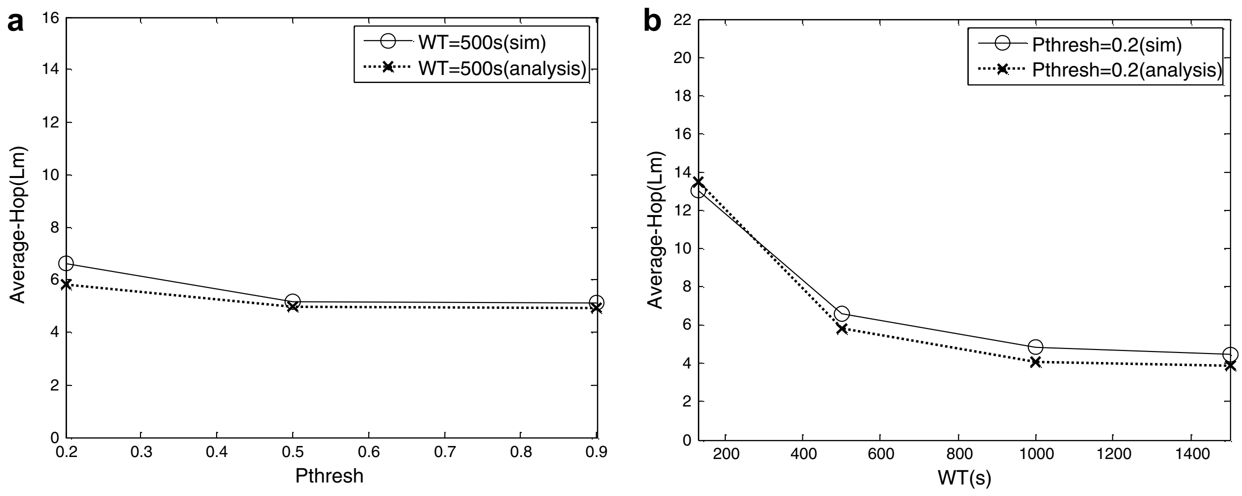


Fig. 6. L_m obtained from analysis and simulations.

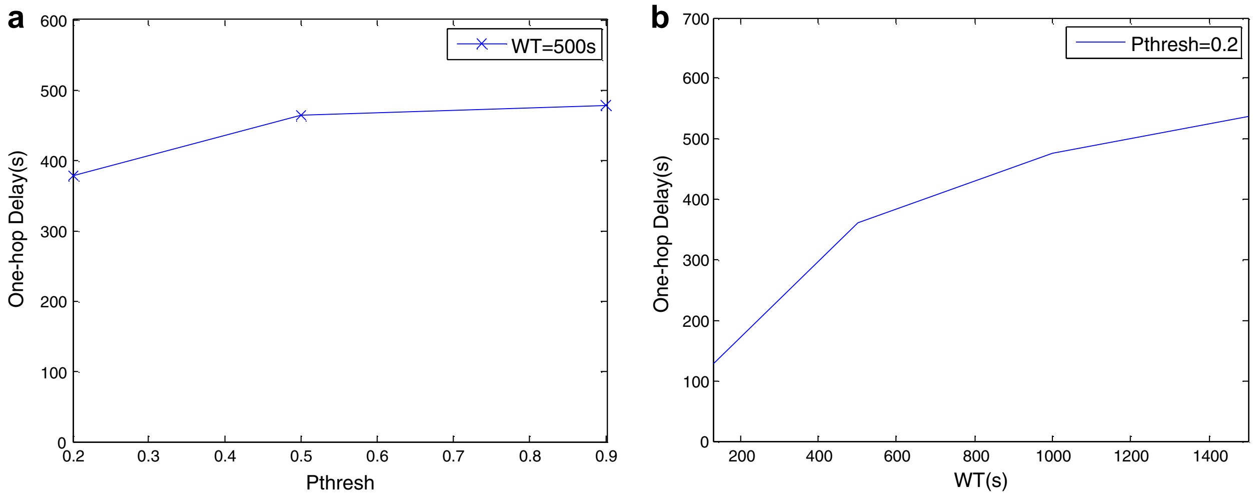


Fig. 7. Derived d_{hop} value for different P_{thresh} or WT values.

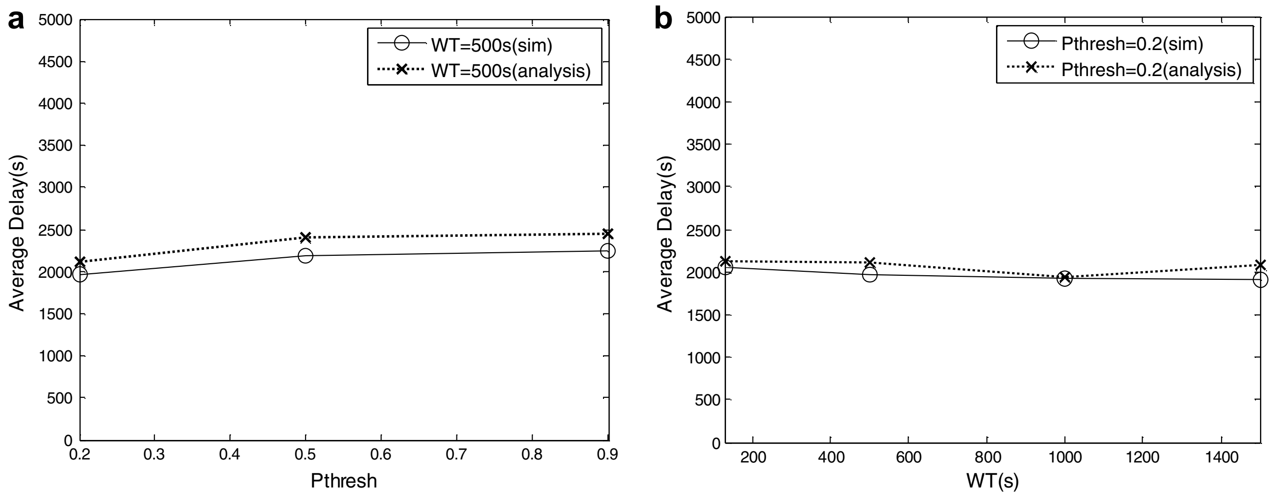


Fig. 8. Avg E2E delay with different P_{thresh} and WT values.

conduct some simulation studies to compare our analytical results with simulation results. We use the same network scenario used for Fig. 6 (i.e. 40 nodes over $3000 \times 3000\ \text{m}^2$). We use two multicast sessions, each with one source and four receivers. Each source generates 0.1 msg/s. The source and the receivers of each session are randomly chosen among the 40 nodes. For each simulation run, we use a warmup period of 1000 s before each flow starts generating traffic for 2000 s, and the simulation continues to run until 10,000 s. In Fig. 8(a) and (b), we plot the average end-to-end delay we obtained from simulations versus the computed $L_m{}^*d_{hop}$ value. The analytical results indicate that when *WT* increases, the average number of hops reduces but the average per-hop waiting time increases. The increasing per-hop waiting time with larger *WT* values means more buffers will be used. Thus, a tradeoff needs to be made. Given a P_{thresh} value, our analysis allows us to select a suitable *WT* value such that the EBMR scheme can achieve small L_m and per-hop delay values.

4.3. Sensitivity analysis of P_{thresh} and WT

Next, we perform a set of experiments to see how different P_{thresh} and *WT* values affect the delivery performance. In these experiments, we use 40 nodes distributed over $3000 \times 3000\ \text{m}^2$. We use two multicast sessions with each session having one multicast source and four receivers. Each multicast source generates 1 bundle every 10 s. The results for the delivery ratio, the average delivery latency, the average data efficiency, the average number of hops, and the average number of queued messages are plotted in Figs. 9a–9d, a 9e respectively. From Fig. 9b, we see that the average delay is the lowest with $P_{thresh} = 0.2$. Our analysis shows that the average numbers of hops for $P_{thresh} = 0.2$, 0.5 (with *WT* = 500 s) are 6 and 5, respectively, and that the average per-hop delay for $P_{thresh} = 0.2$, 0.5 is 340 and 450 s, respectively. Thus, the average E2E delay from analysis for $P_{thresh} = 0.2$, 0.5 is 2040 and 2250 s, respectively. These values match closely with the observed average from the simulations as shown in Fig. 9b. The paired *t*-test [22] for the delivery ratio results with $P_{thresh} = 0.2$ and $P_{thresh} = 0.5$ shows that the difference is statistically significant. Similarly, the paired *t*-test for the average delay results with $P_{thresh} = 0.2$ and $P_{thresh} = 0.5$ shows that the difference is very significant. The paired *t*-test for the delivery ratio and the average delay results with $P_{thresh} = 0.5$ and $P_{thresh} = 0.9$ do not show that their differences are significant.

From the plots, we see that setting $P_{thresh} = 0.2$ gives a slightly higher delivery ratio, low average delay but also lower data efficiency, and higher buffer usage. With increasing *WT* values, the average delay and the data efficiency improves since the nodes can find better next-hop node, and hence the average number of

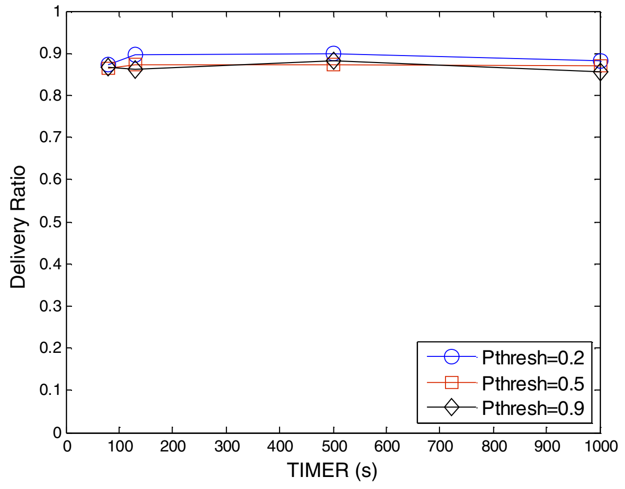


Fig. 9a. Average delivery ratio vs timer.

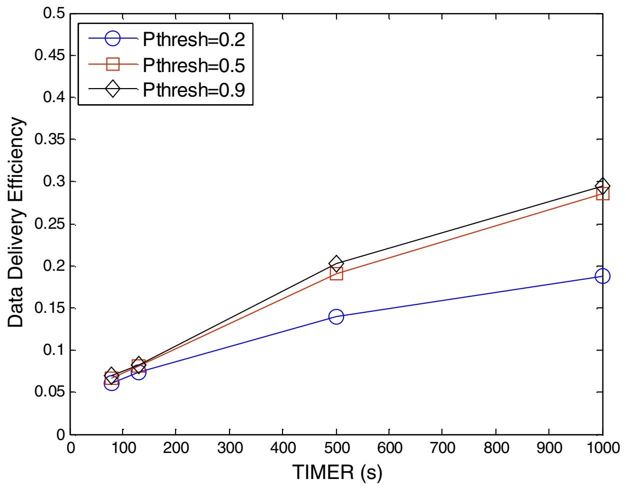


Fig. 9c. Average data efficiency vs timer.

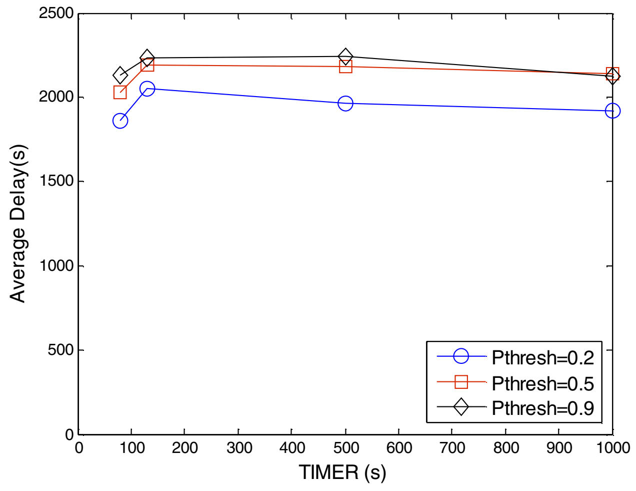


Fig. 9b. Average delay vs timer.

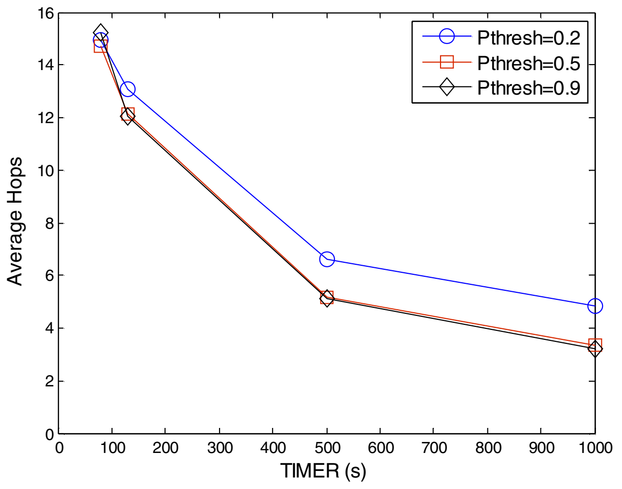


Fig. 9d. Average number of hops vs timer.

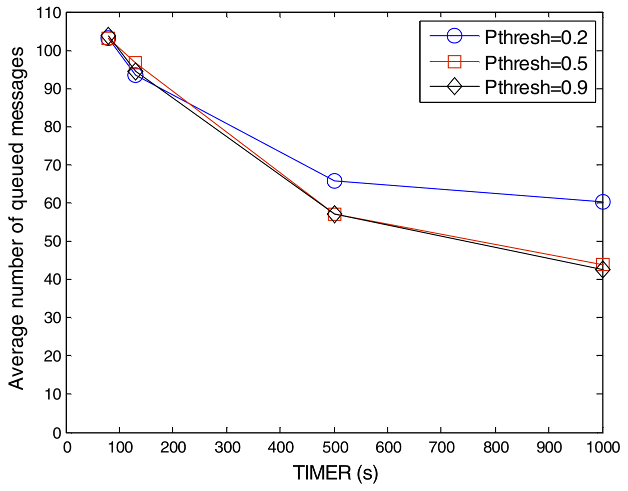


Fig. 9e. Average number of queued messages vs timer.

hops to reach a multicast destination decreases with increasing *WT*. From the results, it seems that setting $P_{thresh} = 0.5$ and $WT = 500$ s is a good trade off between achieving high delivery ratio, high data efficiency, and low buffer usage.

5. Simulation studies

5.1. Simulation setup

In order to evaluate the EBMR scheme, we implement this scheme using NS-2 simulator [17]. In our simulation, the default network scenario is one where 40 nodes are randomly distributed over 3000×3000 m^2. We assume that a 802.11 radio is used in each node. Thus, the default 802.11 parameters are used and the transmission range is set to 250 m.

5.1.1. Mobility model

By default, the nodes move according to the random waypoint model [16]. In the RWP model, each node randomly selects a destination location within the simulated geographical area, and move towards it using a constant speed v. Once it reaches the destination, it will pause for a certain time, then it repeats its action (i.e. picks another destination to move to). Unless otherwise stated, the pause time is set to 0 in our simulations and the node speed is chosen randomly between (1,5) m/s. Another mobility model we use is the Zebranet model. For the Zebranet model, we create a semi-synthentic Zebranet mobility model as follows: we synthesize node speed and turn angle distributions from the Zebranet trace [11], and create other node-movements using the same distribution. We use both distance and time scaling to fit the original data found in the trace into the network environment that we are interested in. For the UMassDieselNet model [21], we extract the locations of 20 buses at different times in one trace, and scale their relative locations to fit into the 3000×3000 m^2. This will be used to represent the node movement. We repeat this process five times using different traces for the five runs. The reason why only twenty buses are used for the UMassDieselNet model is because the number of buses on the road at any time varies and at most we can find twenty different buses that are active simultaneously for the 10,000 s duration that we want for each of our simulation runs.

5.1.2. Traffic model

For the traffic model, we sometimes use (a) one multicast session with one source and multiple receivers, (b) multiple multicast sessions with one source and multiple receivers in each session. Each multicast source generates CBR traffic with a packet size of 512 bytes. We let the source generates traffic after 1000 s of warming up period and the traffic generation lasts for 2000 s but the simulation will run until 10,000 s. Each reported data point is the average of 10 runs.

5.1.3. Performance metrics

The performance metrics we used in our evaluation are:

(1) *Delivery ratio*, which is the ratio of the number of endpoints that receive a message and the number of intended receivers of the message [13]. This metric measures how successful the routing algorithm is in delivering the messages. For example, let *NR* be the number of receivers, s_i be the number of receivers that receive a message *i*, and *T* be the total number of multicast messages sent by the source, then $DR = \sum s_i/(T * NR)$.
(2) *Average delay*, which is defined as the average end-to-end delay incurred by the delivered messages.
(3) *Data efficiency*, which is the total number of messages received divided by the number of transmissions used to deliver such messages [13].
(4) *Average buffer usage*, which is the average number of queued messages per node.

For subsequent subsections, we first present a comparison of the delivery performance achieved by different multicast routing strategies. Then, we study how different mobility models affect the delivery performance of EBMR. Next, we study how the group size (i.e. the number of receivers) and different number of groups affect the delivery performance of EBMR. Last but not least, we study how the node speed affects the delivery performance of EBMR.

5.2. Comparison of different multicast routing schemes

In this section, we compare the following multicast delivery schemes: (a) brute-force unicast delivery using the original Prophet scheme where the source duplicates one message for each multicast receiver and each message is delivered using the original Prophet, (b) the early-decision based multicast strategy (denoted as Multicast), (c) EBMR delivery with $K = 1$, $P_{threshold} = 0.5$ and $WT = 500$ s (denoted as MK = 1), (d) EBMR delivery with $K = 2$, $P(_{threshold} = 0.5$ and $WT = 500$ s (denoted as MK = 2). We use the default network scenario and let the nodes move according to the

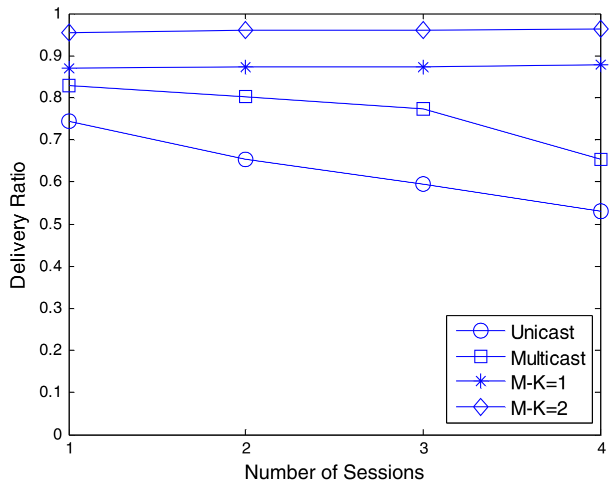


Fig. 10a. Delivery ratio vs number of sessions.

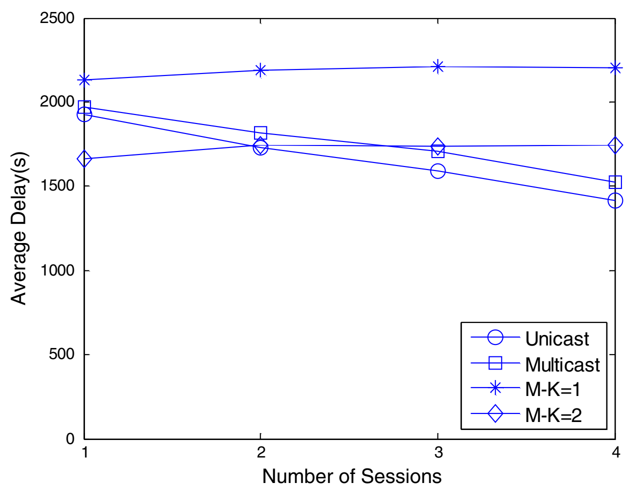


Fig. 10b. Average delay vs number of sessions.

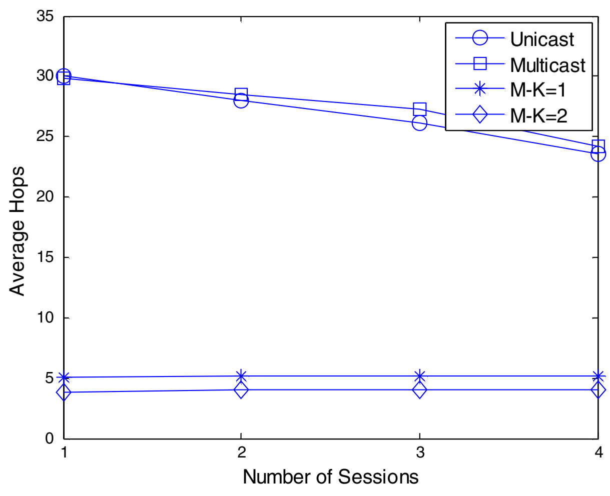


Fig. 10c. Average hops vs average number of sessions.

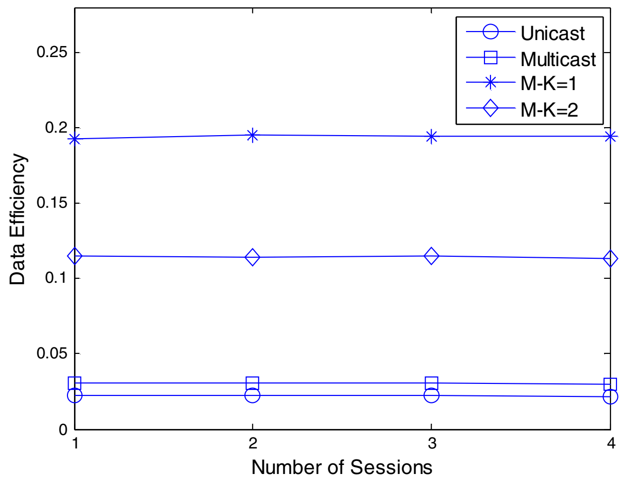


Fig. 10d. Average data efficiency vs number of sessions.

RWP model. Each multicast session has one source and four receivers. A multicast source generates 0.1 bundle every second. We vary the number of multicast sessions in this experiment.

The results for the delivery ratio, the average delay, average number of hops it takes to reach a receiver, and the data efficiency are plotted in Figs. 10a–10c and 10d, respectively. Fig. 10a shows that the MK = 2 scheme achieves the best delivery performance: it has the highest delivery ratio. The plot shows that using unicast approach, the delivery ratio drops with increasing number of sessions. This can be explained as follows: for each session, the source node duplicates four copies per bundle so the total message rate is 0.4 bundle/s. As the number of sessions increases, the source nodes may also be used to relay traffic from other sessions. That increases further the total message rate that these nodes need to transmit. With an IFQ buffer size of 100, many packets will be lost due to IFQ buffer overflows. Thus, the delivery ratio for the unicast approach drops with increasing number of sessions. For other schemes, the total message rate that each source node needs to send out is in the range of 0.1–0.2 bundle/s and hence we see very few IFQ buffer overflows even with increasing number of sessions.

The lower average delay for the unicast and multicast approaches in Fig. 10b is misleading since their delivery ratios drop significantly with increasing number of sessions, and only messages that take fewer hops can be delivered. The average delay for the MK = 1 scheme is higher than that for the Multicast scheme even though the MK = 1 scheme takes only five hops because the messages are queued longer at each intermediate node while waiting for better next-hop node when the MK = 1 scheme is used. The MK = 2 scheme achieves at least 25–30% lower average delay compared to the MK = 1 scheme because its average delay is the average delay of the first message copies that arrive at the receivers. Fig. 10c shows that the average number of hops taken to reach the multicast receivers is significantly reduced with our enhanced EBMR scheme. Since $K = 2$ incurs extra data redundancy, it is not surprising that the MK = 1 scheme achieves the best data efficiency as shown in Fig. 10d. The MK = 2 scheme still provides higher data efficiency than the Unicast and Multicast schemes.

5.3. Impact of mobility models

In this section, we study the impact of mobility models. In addition, we also compare the EBMR scheme with another scheme we design, namely CAMR scheme. We use the default network scenario and use a multicast session with one source and eight receivers. We select three mobility models, namely (a) the random waypoint (RWP) model, and (b) the Zebranet model [11], and UMassBusNet model [21].

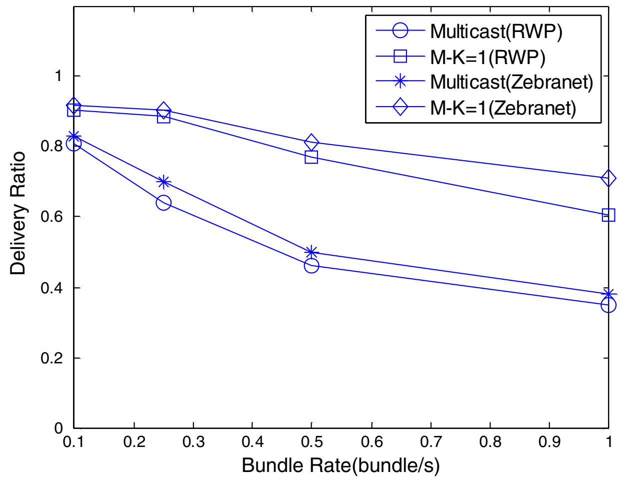


Fig. 11a. Delivery ratio vs message rate.

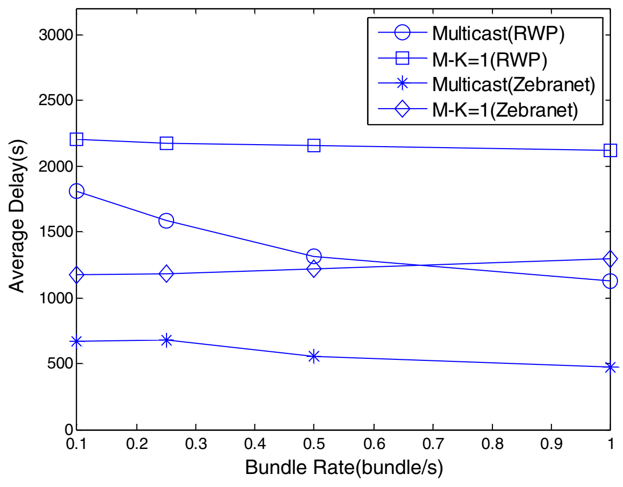


Fig. 11b. Average delay vs message rate.

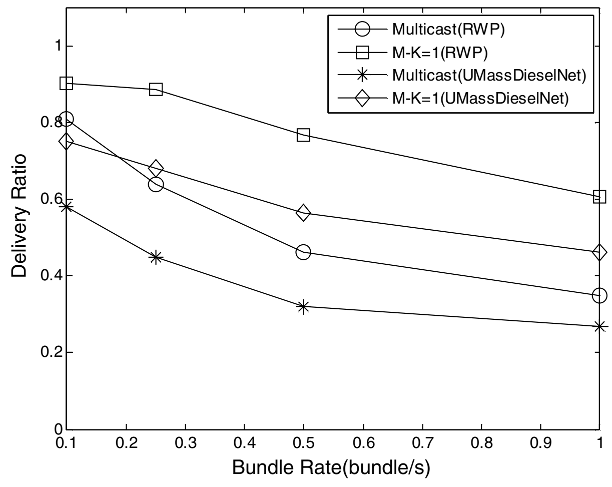


Fig. 12a. Delivery ratio vs bundle rate.

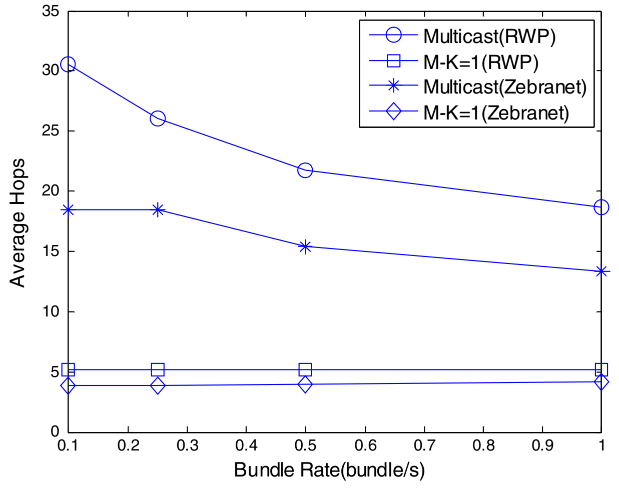


Fig. 11c. Average number of hops vs message rate.

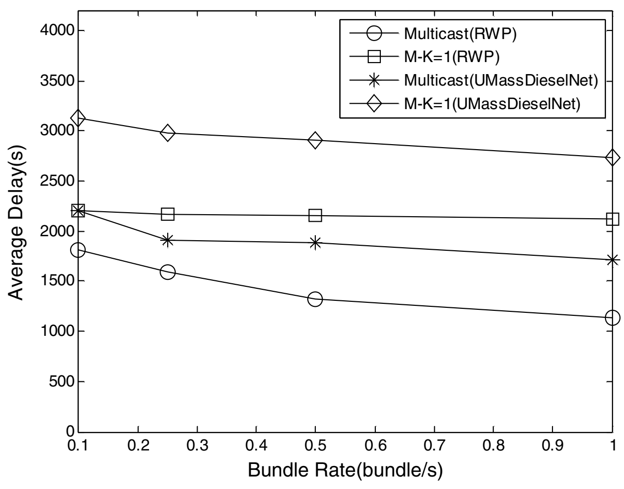


Fig. 12b. Average delay vs bundle rate.

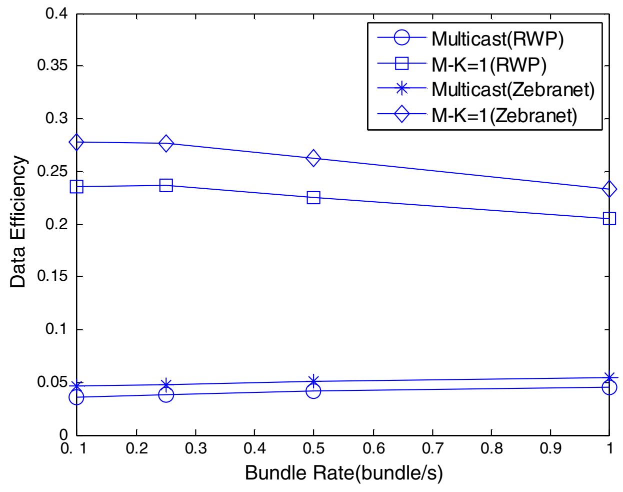


Fig. 11d. Data efficiency vs message rate.

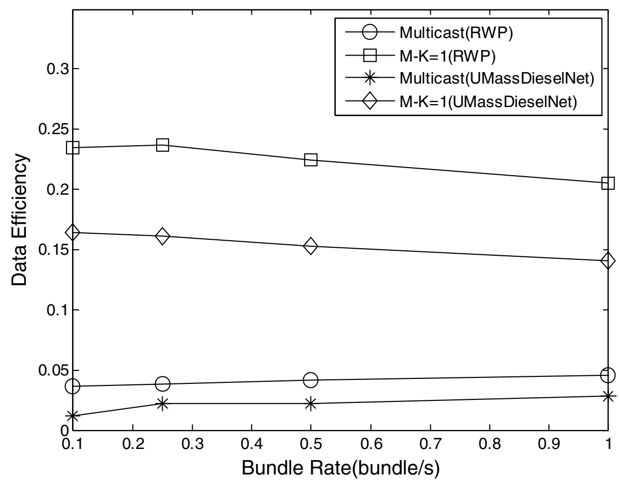


Fig. 12c. Average data efficiency vs bundle rate.

For EBMR, we use the single copy version i.e. MK = 1, and set $P_{thresh} = 0.5$ and $WT = 500$ s. We use one multicast session with eight receivers and vary the source rate from 0.1 to 1 bundle/s. Our results for the delivery ratio, the average delay, the average number of hops, and the data efficiency for the RWP and Zebranet models are plotted in Figs. 11a–11c, a 11d, respectively. Our results for the delivery ratio, the average delay, and the data efficiency for the RWP and UMassDieselNet models are plotted in Figs. 12a, 12b, and 12c, respectively.

There are several observations we made based on our simulation results. First, the delivery ratio drops with increasing message rate. Such drop is due to increasing IFQ overflows as bundle rate increases. We anticipate that if not all queued packets are immediately delivered to the MAC layer when a suitable contact is found but rather spread over a bigger time window (but still within the contact duration), then, the drop can be reduced. We include the results we obtained with such enhancement in Appendix 1. Second, the delivery performance is slightly better with the Zebranet model than with the RWP model irrespective of whether EBMR or Multicast scheme is used. The delivery ratio achieved using the Zebranet model is higher than that achieved using the RWP model. This difference becomes more apparent with increasing multicast source rate as seen in Fig. 11a. The average delay using the Zebranet model is also lower than that using the RWP model (see Fig. 11b). The increased delivery ratio and the lower average delay using the Zebranet model can be attributed to the higher average node speed in the Zebranet model (6 m/s) compared to that in the RWP model (3 m/s). The higher average node speed means shorter mean intercontact time between nodes. As long as the contact duration for the Zebranet model is large enough to allow the two nodes in contact with each other to transfer over relevant queued bundles, the delivery performance will improve.

Our third observation is that the delivery ratio using the EBMR scheme is higher than what is achieved using the multicast scheme because the EBMR scheme attempts to find a better forwarding node by not forwarding messages immediately. The EBMR scheme is able to deliver messages using much fewer hops than the multicast scheme as can be seen in Fig. 11c. The price to pay is a higher average delay as seen in Fig. 11b. The fewer number of hops taken by the EBMR scheme to deliver the messages also results in the EBMR scheme achieving higher data efficiency (shown in Fig. 11d) than the multicast scheme.

Fourth, the delivery performance for UMassDieselNet is the worst among the results obtained from the three mobility models irrespective of whether the EBMR or the multicast scheme is used. The delivery ratio achieved using the UMassDieselNet model (see Fig. 12a) is the smallest while its average delay is the highest amongst the values obtained using the three mobility models (see Fig. 12b). The delivery ratio achieved using the UMassDieselNet model is only 75% at 0.1 bundle/s and 46% at 1 bundle/s compared to 90% at 0.1 bundle/s and 61% at 1 bundle/s when the RWP model is used. This is because the mean intercontact time for the UMassDieselNet is sometimes five times longer than that seen in the RWP model. This causes messages to be queued longer in the nodes and hence the chances of new messages being dropped increase due to buffer overflows.

From Fig. 12b, we observe that the average delay drops slightly with increasing traffic load. This is quite misleading and is caused by the fact that packets that require more hops to be delivered are dropped by the two multicast schemes we consider. We verify this from our simulation results since the average number of hops of successfully delivered packets drops with increasing traffic load.

The data efficiency with the UMassDieselNet model for the EBMR scheme is lower than that achieved with the RWP model. The average number of hops taken by the EBMR scheme to deliver a bundle with the UMassDieselNet model is about 6.7 at 0.1 bundle/s and 5.5 at 1 bundle/s which is higher than what we see in the RWP model.

To improve the delivery performance when the UMassDieselNet model is used, one can use multiple copy approach. We repeat the above experiment with the UMassDieselNet mobility model using $K = 2$ and plot the delivery ratio, the average delay and the data efficiency results for MK = 1, and MK = 2 in Figs. 12d, 12e and

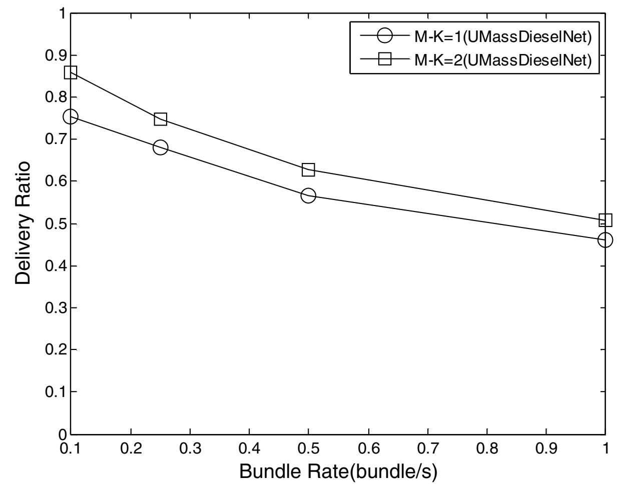


Fig. 12d. Delivery ratio vs bunlde rate (multiple copy).

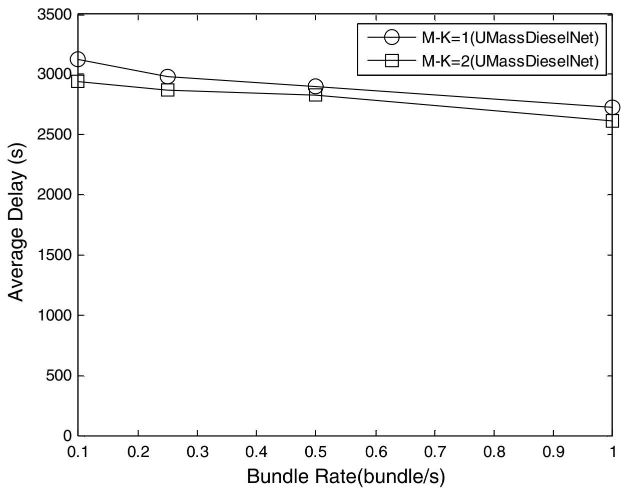


Fig. 12e. Average delay vs bundle rate (multiple copy).

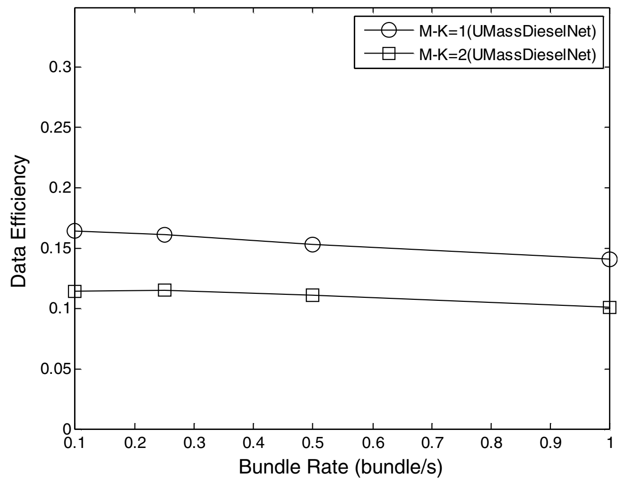


Fig. 12f. Data efficiency vs bundle rate (multiple copy).

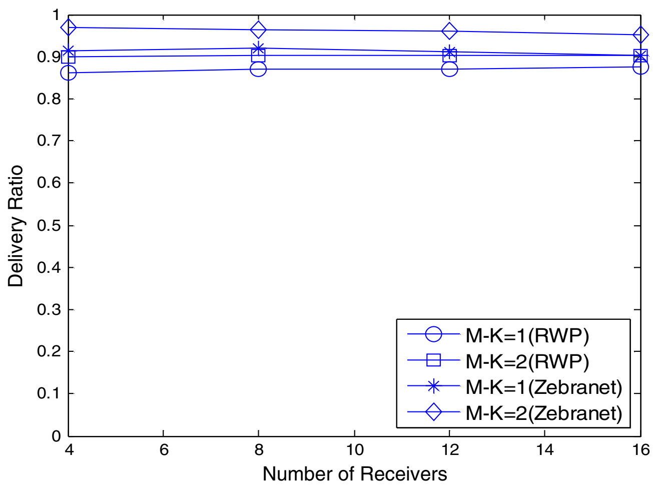


Fig. 13a. Delivery ratio vs number of receivers.

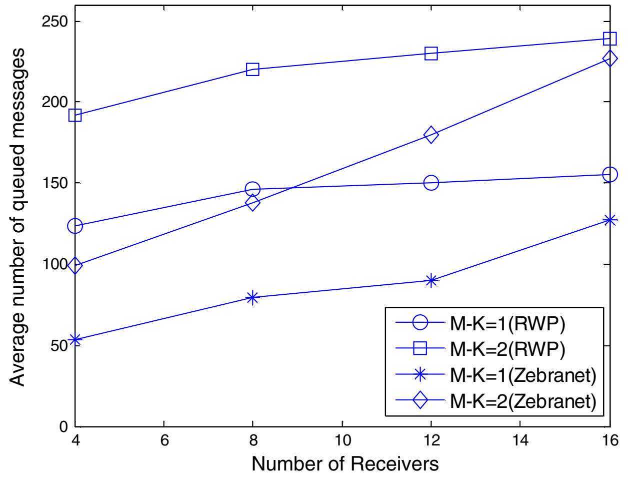


Fig. 13d. Average buffer usage vs number of receivers.

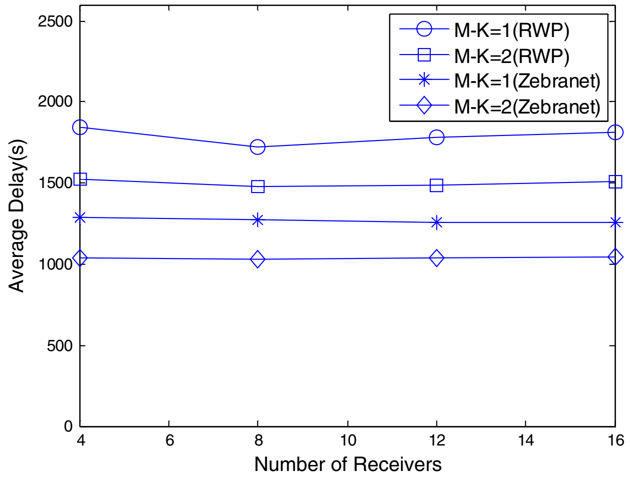


Fig. 13b. Average delay vs number of receivers.

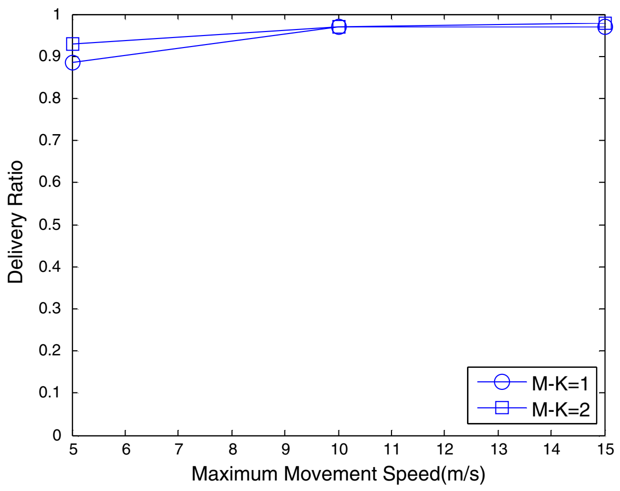


Fig. 14a. Delivery ratio vs max node speed.

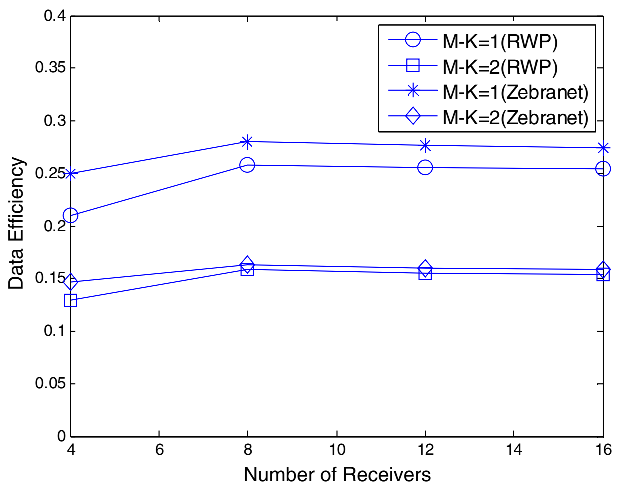


Fig. 13c. Average data efficiency vs number of receivers.

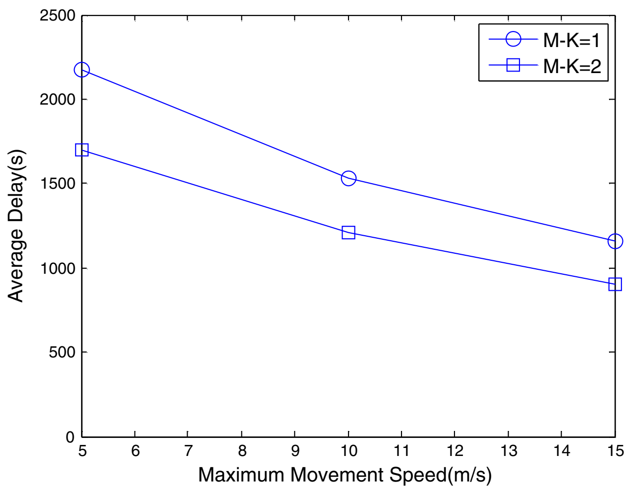


Fig. 14b. Average delay vs max node speed.

12f, respectively. The results obtained indicate that the deliver ratio improves by 8.7% (at 1 bundle/s) to 13.3% (0.1 bundle/rate) when $K = 2$ is used. The price to pay is a 30–40% reduction in the data efficiency.

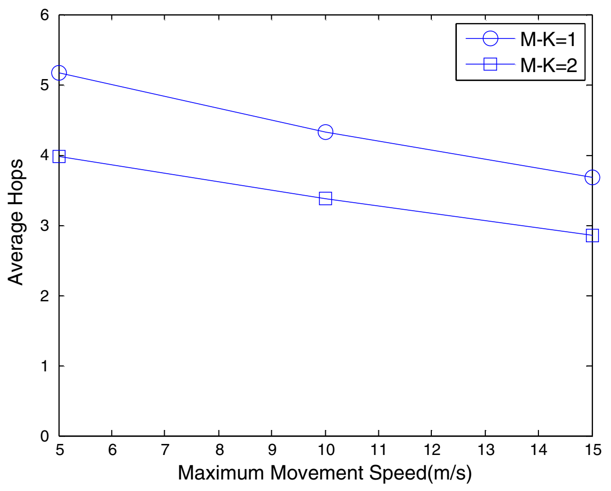


Fig. 14c. Average hops vs max node speed.

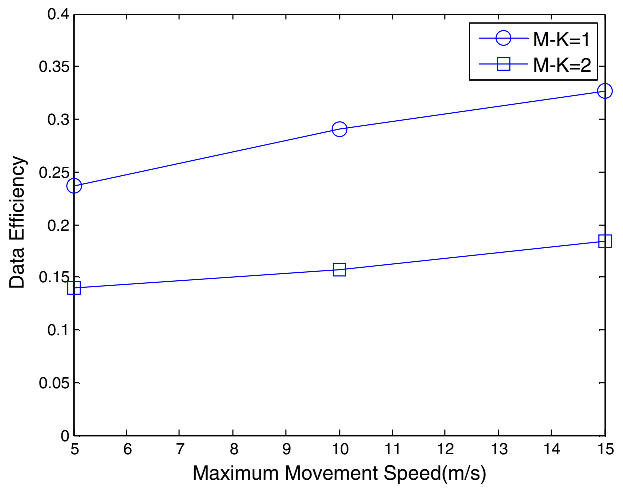


Fig. 14d. Data efficiency vs max node speed.

5.4. Impact of different number of receivers

In this section, we perform a set of experiments with one multicast session having a single source but the number of receivers is varied. The source generates messages at a rate of 0.25 bundle/s. We use the default network scenario. The nodes either move using RWP or Zebranet model. We use the EBMR scheme with $K = 1$ and $K = 2$. Figs. 13a, 13b, 13c, 13d plot the average delay ratio, the average delivery latency, the average data efficiency, and the average buffer usage results we obtain in this experiment, respectively.

From the plots, one can see that the delivery ratio and the average delay remains almost the same with different numbers of receivers. The performance of the EBMR scheme does not degrade with increasing number of receivers because the network is very sparse and the multicast source rate is not high enough to cause frequent buffer overflows. The MK = 2 scheme provides better delivery performance (higher delivery ratio and lower average delay) because extra copies explore path diversity. The price to pay for this better delivery performance is lower data efficiency (see Fig. 13c) and higher buffer usage (see Fig. 13d).

Another observation is that the delivery performance for Zebranet model (higher delivery ratio, lower average delay and lower buffer usage) is better than that for RWP model. As explained earlier, this may be due to the fact that the nodes in the Zebranet model move faster and hence the chances for meeting the receivers increase.

The data efficiency increases initially with the number of receivers since the chances of using the same next-hop node to deliver multicast messages increase as the number of receivers increases. Beyond a certain point, no such gain can be observed so the average data efficiency then remains almost the same. The average buffer usage increases with increasing number of receivers.

5.5. Impact of node speed

Last but not least, we investigate the impact of node speed on the delivery performance. We use a multicast group with one source and four receivers. The source generates 0.25 bundle/s. The nodes move according to the RWP mobility model. We use both MK = 1 and MK = 2 schemes. Figs. 14(a) to 14(d) plot the results we obtain for the delivery ratio, the average delay, the average hops and the data efficiency respectively.

Our results indicate that as the maximum node speed increases, the delivery performance improves (slightly higher delivery ratio, lower average delay, slightly higher data efficiency). The analytical results obtained using higher speed values match well with the observed simulated values, e.g. with an average speed of 6 m/s (corresponding to max speed of 11 m/s), the average hop from our analysis is 4.1 (observed value is 4.2) and the average delay from analysis is 1300 s (observed value is 1400 s). The slightly higher data efficiency is due to a smaller average number of hops taken to reach the receivers when the maximum node speed increases. Again, we see that MK = 2 produces lower average delay because the extra copies explore path diversity and hence can reach the destinations faster. The price to pay for the MK = 2 scheme is a lower data efficiency.

6. Conclusion

In this paper, we have presented an encounter-based multicast routing scheme for DTNs. Our EBMR scheme allows nodes to cache the data until a good next-hop node can be found to relay the messages to the destinations. Via analysis, we have shown that with appropriate choice of P_{thresh} and WT, our EBMR scheme takes fewer number of hops to reach a destination. Via simulation studies, we have demonstrated that this scheme can achieve high delivery ratio with reasonable data efficiency over different scenarios. There are several interesting issues we intend to explore further, e.g. we wish to investigate the impact of other mobility models, e.g. the RPGM [20], traffic patterns, and power management on the multicast delivery performance. In addition, we intend to evaluate the EBMR scheme in scenarios with non-homogenous node distribution. We also intend to explore an adaptive scheme where data replication is invoked only when the delivery predictability is below a certain threshold. We intend to implement this scheme on a medium size testbed to evaluate its performance with real multicast applications running in the DTN nodes.[1]

Acknowledgements

This work has been supported by DARPA under Contract W15P7T-06-C-P430. 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.

[1] Our testbed evaluation results will be released around June or July, 2008.

Table 2a
Performance of MK = 1 scheme without time spreading (as shown in Fig. 11).

Bundle rate (bundle/s)	Delivery ratio	Delay (s)	Ave-hop	Data efficiency	Ave-queued messages
0.1	0.902	2200.9	5.2	0.235	46.4
0.25	0.886	2172.5	5.2	0.236	105.1
0.5	0.768	2156.3	5.2	0.225	174.9
1	0.606	2117.7	5.2	0.205	269.1

Table 2b
Performance of MK = 1 scheme with time spreading.

Bundle rate (bundle/s)	Delivery ratio	Delay (s)	Ave-hop	Data efficiency	Ave-queued messages
0.1	0.926	2190.6	5.2	0.235	44.7
0.25	0.924	2178.9	5.2	0.233	109.4
0.5	0.893	2144.9	5.1	0.232	202.8
1	0.746	2157.8	5.2	0.217	333.5

Table 2c
Performance of MK = 1, MK = 2 with/without time spreading.

	DR	Average delay (s)	Data efficiency
MK = 1	0.46	2732.2	0.140
MK = 1 (spread)	0.50	2740.8	0.148
MK = 2	0.50	2612.5	0.101
MK = 2 (spread)	0.56	2636.1	0.107

Appendix A

Here, we present simulation results we obtained with the same scenario for Figs. 11a–11d, where MK = 1 scheme with Pth = 0.5, WT = 500 s is used. In this scenario, one multicast group with eight receivers is used and the bundle rate is varied from 0.1 to 1 bundle/s. The mobility model used is RWP.

As explained in the text, when a node meets another node that is suitable to be a forwarding node, instead of immediately forwarding all queued packets into the IFQ layer, the time spreading feature forwards the queued packets into the IFQ layer over a period of time (set to 1 s in the results we presented). Such time spreading feature avoids some IFQ overflows and hence improve the delivery performance. We have seen similar improvements for Zebranet model as well. The conclusions we draw in Section IV.D that the delivery performance obtained using Zebranet model is better remain unchanged since this time spreading feature provides better delivery performance for all mobility models (see Tables 2a, 2b).

We also repeated the MK = 1 and MK = 2 experiments we did using the UMassBusNet trace (Figs. 12d–12f) using the time spreading feature to see how much improvement we can get. Table 2c tabulates the results we obtained for the case with the multicast bundle rate set at 1 bundle/s.

References

[1] D. Johnson, D. Maltz, Dynamic source routing in ad-hoc wireless neteworks, in: Proceedings of ACM Sigcomm, August, 1996.
[2] S. Das, C. Perkins, E. Royer, Performance comparision of two on-demand routing protocols, in: Proceedings of IEEE Infocom, March, 2000.
[3] E. Royer, C. Perkins, Multicast adhoc on-demand (MAODV) routing, draft-ietf-manet-maodv-00.txt, July, 2000.
[4] J.J. Garcia-Luna-Aceves, A multicast routing protocol for adhoc neworks (CAMP), in: Proceedings of IEEE Infocom, 1999, pp. 784–792.
[5] Forrest Warthman, Delay-tolerant networks (DTNs) – a tutorial, based on DTN Research Group Internet Draft, March, 2003.
[6] K. Fall, A delay-tolerant network architecture for challenged internets, in: SIGCOMM, 2003.
[7] M.M.B. Tariq, M. Ammar, E. Zegura, Message ferry route design for sparse ad hoc networks with mobile nodes, ACM MobiHoc, May 22–25, 2006.
[8] S. Jain, K. Fall, R. Patra, Routing in a delay tolerant network, SIGCOMM'04, August 30 – September 3, 2004.
[9] J. Burgess, B. Gallagher, D. Jensen, B.L. Levine, Maxprop: routing for vehicle-based disruption-tolerant networks, in: INFOCOM, 2006.
[10] A. Lindgren, A. Doria, O. Scheln, Probabilistic routing in intermittently connected networks, in: Proceedings of the Workshop on Service Assurance with Partial and Intermittent Resources, August, 2004.
[11] Y. Wang, S. Jan, M. Martonosi, K. Fall, Erasure-coding based routing for opportunistic networks, in: Proceedings of ACM Sigcomm WDTN Workshop, August, 2005.
[12] S. Jain, M. Demmer, R. Patra, K. Fall, Using Redundancy to cope with failures in a delay tolerant network, in: Proceedings of ACM Sigcomm, August, 2005.
[13] W. Zhao, M. Ammar, E. Zegura,Multicasting in delay tolerant networks: semnatics models and routing algorithms, in: Proceedings of ACM Workshop on Wireless DTN, August, 2005.
[14] Q. Ye, L. Cheng, M. Chuah, B. Davison, On-demand situation-ware multicasting in DTNs, in: Proceedings of IEEE Vehicular Technology Conference, May, 2006.
[15] P. Yang, M. Chuah, Context-aware multicast routing schemes for DTNs, in: Proceedings of ACM Workshop on PE-WASUN, August, 2006.
[16] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Communications & Mobile Computing (WCMC) 2 (5) (2002) 483–502.
[17] The network simulator ns-2, Available from: <http://www.isi.edu/nsnam/ns/>.
[18] A. Khelil et al., Contact-based mobility metrics for delay-tolerant ad hoc networking, in: Proceedings of IEEE MASCOTS, 2005.
[19] T. Spyropoulos, Performance analysis of mobility-assisted routing, in: Proceedings of ACM MobiHoc, 2006.
[20] X. Hong, M. Gerla, G. Pei, C.C. Chiang, A group mobility model for ad hoc wireless networks, in: Proceedings of ACM International Workshop on Modeling, Analysis and Simulations of Wireless and Mobile Systems (MSWiM), August, 1999.
[21] X. Zhang, J. Kurose, B. Levine, D. Towsley, H. Zhang, Modeling of a bus-based disruption tolerant netwok trace, in: Proceedings of ACM Mobihoc, 2007.
[22] W. Navidi, Statistics for Engineers and Scientists, McGraw-Hill, New York, 2006.