Routing in Delay Tolerant Networks with Fine-Grained Contact Characterisation and Dynamic Message Replication

Luming Wan, Haibo Zhang, Feiyang Liu, and Yawen Chen
Department of Computer Science, University of Otago, Dunedin 9016, New Zealand
Email:{luming, haibo, feiyang, yawen}@cs.otago.ac.nz

***Abstract*—Pairwise contacts in Delay-Tolerant Networks (DTNs) for applications such as bus or smartphone based social networking commonly show some regular repeating patterns. Most existing routing protocols only implicitly exploit these patterns to predict future contacts. To enhance message delivery rate, most of the schemes allow messages to be replicated and forwarded to encountered nodes. However, there is no efficient mechanism for dynamically controlling message replication to achieve high message delivery rate with very low message overhead. In this paper, we present FGDR, a routing protocol designed for DTNs by leveraging fine-grained contact characterisation and dynamic message replication. In FGDR, the history contact is characterised in a fine-grained manner using a sliding window mechanism, and an up-to-date future contact prediction can be made based on the most recent history data. We design an efficient message replication scheme, in which replication is controlled in a fully decentralised manner by taking into account the expected message delivery rate, the replication history, and the quality of the encountered node. A replica can be generated only when it is necessary to fulfill the expected message delivery rate. We evaluate our scheme through trace-driven simulations, and results show FGDR can achieve much higher message delivery rate with lower message overhead in comparison with existing schemes.**

***Index Terms*—Delay-tolerant networks (DTNs), fine-grained, routing, message replication, contact pattern**

I. Introduction

With the popularity of ubiquitous computing and communication, wireless mobile devices such as smartphones, tablets, and laptops may wish to communicate at any time and in any place, regardless of whether there exists an end-to-end path between the source and the destination. Delay-tolerant networking is an architecture proposed for communication in networks that lack continuous connectivity. DTNs have many applications such as vehicular networks, opportunistic mobile data offloading [1], and social network analysis [2]. However, due to the lack of contemporaneous end-to-end paths, devices need to carry and forward messages opportunistically upon encountering the destinations, or seek other encountered devices to help message delivery. In addition, devices in DTNs generally suffer from constraints in memory, communication bandwidth, and battery power. These special characteristics and constraints make routing in DTNs a challenging problem.

Contacts between devices in DTNs usually exhibit a high degree of repetition due to the daily scheduled routines of the persons who carry these devices [3]. Hence, most existing routing schemes use the history of previous contacts to predict future contacts. However, in most prediction-based routing schemes [4] [5] [6] [7], the contact history for each pair of nodes is compressed into a piece of coarse-grained information called encounter predictability that is used as the routing metric. However, this coarse-grained information represents only the long-term average contact probability, which can not immediately detect and adapt to the contact pattern change. The long responding time will lead to the inaccuracy on predicting future contacts, and consequently degrade the routing performance.

To enhance message delivery rate, most existing routing protocols are replication-based [4] [5] [6] [8], i.e., multiple copies (or replicas) per message are generated and spread to the network with the expectation that at least one replica will successfully reach the destination. In PRoPHET [4], a message replica is generated for any encountered node that has a larger probability to contact the message's destination. In *Spray and Wait* [9], a fixed number of message replicas per message are initially generated by the source and spread to the network. Spreading a small number of replicas may not achieve the best message delivery rate, but over-spreading may also degrade the routing performance due to message dropping caused by message buffer overflow. A key challenge for replication-based routing schemes is to determine the optimal number of replicas per message that need to be generated and spread.

In this paper, we present FGDR, a routing scheme designed for DTNs by taking the advantages of **F**ine-**G**rained contact characterisation and **D**ynamic message **R**eplication. The main contributions of this paper are summarised as follows:

- We proposed a memory-efficient scheme to characterise the pairwise contact pattern based on bit matrix. A slotted sliding window mechanism is employed to extract fine-grained information from the contact history and store them in the bit matrix. It is able to compute the encounter probability between two nodes *within only the life time of a given message*, which can immediately detect and adapt to any contact pattern change.
- We proposed an efficient mechanism to control message replication. Each message in our scheme is associated with an expected delivery rate. Message replication is performed dynamically in a decentralised manner to fulfill the expected message delivery rate,
- We evaluated the performance of FGDR through trace-driven simulations. Simulation results show that: (1) contact prediction based on fine-grained contact history

978-1-5090-2185-7/16/$31.00

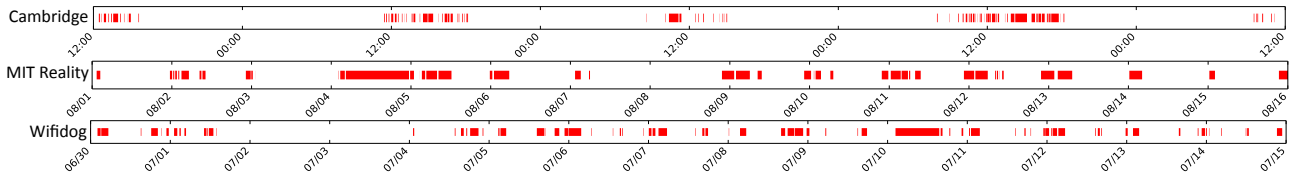

Fig. 1. Contacts between two randomly chosen devices in the Cambridge, MIT Reality, and WiFiDog traces

can greatly improve message delivery rate when contacts between nodes show a high degree of repetition; (2) the dynamic message replication mechanism can significantly enhance message delivery rate with extremely low message overhead in comparison with existing schemes.

The rest of this paper is organized as follows: Section II gives an overview of FGDR. Section III illustrates the detailed implementation of FGDR, including the representation of fine-grained contact history, contact prediction, and dynamic message replication scheme. Section IV evaluates the performance of our scheme by trace-driven simulations, and Section V concludes the paper.

II. Motivation and Overview

A. Motivation

From our observation and also as mentioned in several literatures [2] [10], the daily social activities and communications often exhibit a very high degree of repetition. Fig. 1 shows the pairwise contacts between two active devices randomly chosen from three data traces: Cambridge [11], MIT Reality [12], and WiFiDog [13]. It can be seen that most contacts occurred roughly in the same period of a day. Many of the existing routing protocols leverage such contact regularity to enhance their contact prediction. However, most of them utilise only a piece of coarse-grained information for the prediction. Such long-term average estimated predictability cannot immediately adapt to any contact regularity change, thus that will bring inaccuracy to the prediction of the near future. Because of this, we decide to characterise the history contacts into a fine-grained manner, by taking into account the contact time between two nodes. The encounter probability for only a short time, e.g., the lifetime of a packet, can be estimated based on a particular period of historical information from the fine-grained contact history.

Memory limitation is usually a critical issue for DTN environment. Many routing protocols deploy multi-message-copies mechanism to ensure the delivery rate as well as the reliability of their routing prediction. However, as most existing protocols do not have efficient stopping policy for the message replication, they usually suffer a serious memory overflow issue because of the massive message replicas. The overall aim of DTN routing protocols is to reduce the message overhead as much as possible, while still securing a high message delivery rate. Our proposed message replication control scheme enables a node to keep learning the network from each time it communicates with others. A node can easily find out by itself whether a message already has enough replicas in the network to be delivered to the destination.

B. Protocol Overview

We consider unicast messages, where each message is characterised by a source address, a destination address, a delivery deadline, and an expected delivery rate imposed by either the user or the upper-layer application or set with a default value. Each node uses *time-slotted sliding windows* to maintain the most recent contact history, with the specified contact time , as shown in Fig. 2. The sliding window is divided into slots with equal length (e.g., 1 hour), and the length of the slots controls the granularity of the contact history. Each node keeps track of the nodes it contacted in each time slot. In such a way the encounter predictability can be computed based on the most recent history, rather than a long term average estimation. As illustrated in Fig. 2, node a generates a message at $8am$, and needs to deliver it to node b within 3 hours. The probability for node a to directly contact node b in the following 3 hours can be estimated using only the past contacts occurred between $8am$ and $11am$, thereby improving the accuracy of contact prediction. Moreover, the sliding window mechanism can quickly respond to the changes on contact patterns by incorporating new contacts and removing the outdated ones, thereby overcoming the drawback of the aging mechanism.

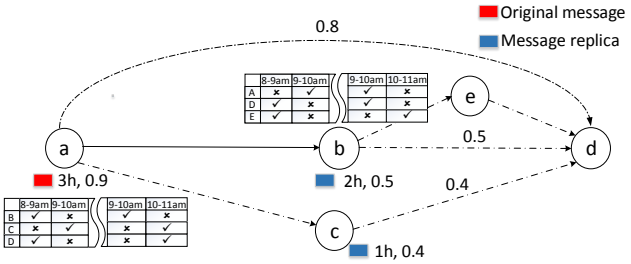

Fig. 2. Message replication and forwarding

To reduce message overhead, message replication and forwarding in our scheme is driven by the message's expected delivery rate in a fully decentralised manner. Each message can generate multiple replicas, and each replica can further generate more replicas to fulfill the message's expected delivery rate. The expected delivery rate associated with each replica represents the accumulative probability that is expected to be achieved by this replica and all its descendants that are either directly or indirectly generated based on this replica. When node a that carries a replica encounters node b, node a stops spreading new replicas if the expected delivery rate of the carried replica has been fulfilled; otherwise it decides whether a new replica needs to be forwarded to node b and how much delivery rate this new replica is expected to achieve, based on the probability for node b to contact the message's destination.

An Illustrating Example: as shown in Fig. 2, node a has a message to be delivered to node d in 3 hours with an expected delivery rate of 0.9. According to the contact history in $a's$ sliding window, node a has a probability of 0.8 to directly encounter node d within 3 hours. So it keeps a replica for this direct delivery. After one hour, node a encounters node b and learns that node b has a probability of 0.5 to contact node d. As the expected delivery rate of the original message has not been fulfilled, node a sends a replica to node b. The expected delivery rate for this replica is set to 0.5, and the deadline is set to 2 hours from now. After two hours node a encounters node c and sends another replica to node c. Since then, the expected delivery rate (0.9) has been fulfilled, node a stops spreading more replicas to other encountered nodes. Since the contact probabilities may change over time, each node needs to dynamically control message replication. For example, node b did not encounter node d in the first hour, and the contact probability between b and d drops to 0.2. To fulfil the expected delivery rate of 0.5, node b spreads a new replica to node e.

III. FGDR: Fine-Grained Contact Characterisation and Dynamic Message Replication Control

A. Contact Characterisation and Prediction

Each node maintains a memory-efficient bit matrix to characterise the history contact. Let L_w denote the length of the time-slotted sliding window, and L_s represent the length of each time slot in the window. Each node a maintains a bit matrix, denoted by $\mathcal{M}_a$, to represent the recent contacts occurred between node a and its encountered nodes during the time period covered by sliding window. $\mathcal{M}_a$ is defined as

$$\mathcal{M}_a[b][k] = \begin{cases} 1, & \text{if } \max_{i=1}^{n_k} l_{ab}^k(i) \geq \alpha \ \& \ \sum_{i=1}^{n_k} l_{ab}^k(i) \geq \beta; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where n_k is the number of contacts occurred between a and b in time slot k, and $l_{ab}^k(i)$ is the contact duration for the i^{th} contact. $\mathcal{M}_a[b][k]$ is marked to 1 only when there is at least one contact in slot k with contact duration no smaller than a threshold α, and the accumulative contact time in time slot k is no smaller than a threshold β. The reason of using two thresholds α and β is to take into account the following scenario: any single contact in slot k is not long enough for the two nodes to transfer all messages that need to be exchanged.

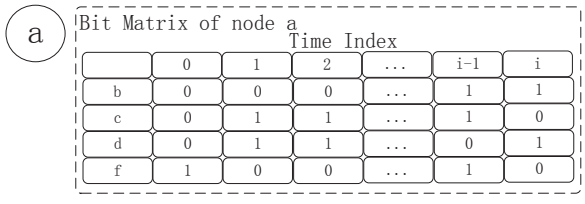

Fig. 3. Fine-grained bit matrix

Fig.3 is an example of the bit matrix held by a node. In the slotted sliding window, the history for one week is represented with $\frac{24\times7}{L_s}$, and the window maintains contact history of $n_w = L_w/\frac{24\times7}{L_s} = \frac{L_w\times L_s}{168}$ weeks. We assume n_w is an integer, i.e., L_w is divisible by $\frac{24\times7}{L_s}$. Given any slot k in the future, the sliding window must contain n_w slots that correspond to the same period of a day as slot k, but in different weeks. The index for such a slot m is $index(m) = (k - m\times\frac{168}{L_s})\ mod\ \frac{L_w}{L_s}$, where $m \in [0, n_w - 1]$, and mod is a Modulo operation that returns positive remainder. Let $p_{ab}(k)$ denote the probability that nodes a and b will contact in slot k. Then $p_{ab}(k)$ can be estimated using the history characterised by these n_w slots as follows:

$$p_{ab}(k) = \frac{\sum_{m=0}^{n_w-1} \mathcal{M}_a[b][index(m)]}{n_w}. \tag{2}$$

Suppose node a carries a message/replica that needs to be delivered to node b during a time period $[t_s, t_e]$. Then $p_{ab}(t_s, t_e)$, the overall probability that covers the entire message lifetime, can be estimated as follows:

$$p_{ab}(t_s, t_e) = 1 - \prod_{i=k}^{k+\frac{t_e-t_s}{L_s}} (1 - p_{ab}(i)). \tag{3}$$

B. Message Replication and Routing

Each message or replica m has a lifetime T_m and an expected delivery rate E_m, where E_m is the accumulative delivery rate to be achieved by this message/replica and all other replicas that are either directly or indirectly generated based on m. Based on E_m, the node that carries m will decide how many replicas should be generated to fulfill E_m. Suppose that k replicas have been generated and forwarded to nodes in $\mathbb{F}_m = \{f_m^1, f_m^2, ..., f_m^k\}$, and the probabilities for these k replicas to reach the destination is given in $\mathbb{P}_m = \{p_m^1, p_m^2, ..., p_m^k\}$. The delivery rate achieved by m and the k replicas, denoted by $P_m(k)$, can be computed as follows:

$$P_m(k) = 1 - \prod_{i=0}^{k} (1 - p_m^i), \tag{4}$$

where p_m^0 is the probability of delivering m from the node that carries m directly to the destination. Note that the computation of $P_m(k)$ does not require the maintenance of either $\mathbb{F}_m$ or $\mathbb{P}_m$ since the production part of Eq. (4) is the probability that all replicas failed to reach the destination, which can be computed incrementally upon the generation of each replica. If $P_m(k) \geq E_m$, the node that carries m will stop generating more replicas.

Algorithms 1 and 2 give the procedures of message replication and forwarding at sender (node a) and receiver (node b), respectively. When node a encounters node b, node a first delivers all its messages that are destined to node b. After that node a checks how node b can help deliver the other messages it carries. To do this, node a generates a summary vector SV_a that contains information of the undelivered messages in the format of $< ID_m, Dest_m, T_m, \gamma_m >$, where ID_m and $Dest_m$ are the message ID and destination, T_m is the message lifetime, and γ_m is the minimum expected delivery rate for a replica of m that is used to eliminate bad forwarders. After generating SV_a, node a sends it to node b. Upon receiving SV_a, for each $< ID_m, Dest_m, T_m, \gamma_m >$, node b will check if it has already carried a replica with ID_m. If not, node b calculates the probability for itself to deliver a replica of m to the destination before the deadline, denoted by $\bar{p}_{bDest_m}$, according to Eq. (3). If $\bar{p}_{bDest_m} \geq \gamma_m$, it indicates that node

Algorithm 1: Replication and Forwarding at Sender

```
/* Upon encountering node b:                                   */
for each message m in a's buffer do
    if Dest_m == b then
        Forward message m to node b;
        Delete message m;
    else
        Add < ID_m, Dest_m, T_m, γ_m > to SV_a;
Send SV_a to node b;
/* Upon receiving ACK for SV_a:                                */
for each < ID_m, p̄_m > in ACK do
    case P_m(k) < E_m & p̄_m ≥ E_m
        Forward message m to b;
    case P_m(k) < E_m & p̄_m < E_m
        Generate a replica of m and forward it to b;
        Update P_m(k);
    case P_m(k) ≥ E_m & p̄_m > p^0_m
        Forward message m to b;
```

Algorithm 2: Replication and Forwarding at Receiver

```
/* Upon receiving SV_a from node a:                            */
for each < ID_m, Dest_m, T_m, γ_m > in SV_a do
    if node b carries a replica with ID_m then
        Continue;
    else
        Calculate p̄_{bDest_m} according to Eq. (3).
        if p̄_{bDest_m} ≥ γ_m then
            Add < ID_m, p̄_{bDest_m} > to the ACK message;
Send ACK to node a;
```

b is a good forwarder for m, and $< ID_m, \bar{p}_{bDest_m} >$ is added to the ACK message. In the end, node b sends the ACK message to node a and informs which messages it chooses to help delivering. After receiving the ACK message, for each $< ID_m, \bar{p}_m >$, node a decides whether to forward message m or send a replica of m to node b using the following rules:

- Rule1: If $P_m(k) < E_m$ & $\bar{p}_m \geq E_m$, node a directly forwards the message/replica m to node b. This is because that the expected delivery rate allocated for m has not been fulfilled, but node b can achieve an expected delivery rate no smaller than E_m if it carries m. Hence, there is no need to generate a new replica.
- Rule2: If $P_m(k) < E_m$ & $\bar{p}_m < E_m$, node a generates a new replica and sends it to node b. The expected delivery rate for the new replica is the probability for node b to contact the destination, and node a updates $P_m(k)$.
- Rule3: If $P_m(k) \geq E_m$ & $\bar{p}_m > p^0_m$, node a forwards m to node b. This is because, even though the expected delivery rate E_m has been fulfilled, node b can provide higher delivery rate than node a.

An Illustrated Example: As shown in Fig. 4(a), node a has a message to be delivered to node d in 9 hours with an expected delivery rate of 0.9. The probability for node a to send the message to node d through direct contact is 0.4. As shown in Fig. 4(b), node a encounters node c after one hour, and forwards a replica to node c according to Rule2. The expected delivery rate for this new replica is set to 0.8. After two hours node a encounters node b and forwards it another replica. After that, the expected delivery rate for the original message at node a has been fulfilled, and node a stops spreading replicas. As shown in Fig. 4(c), node b carries a replica and encounters node e that has a larger delivery rate (0.7) than b. According to Rule3, node b forwards the carried replica to node e.

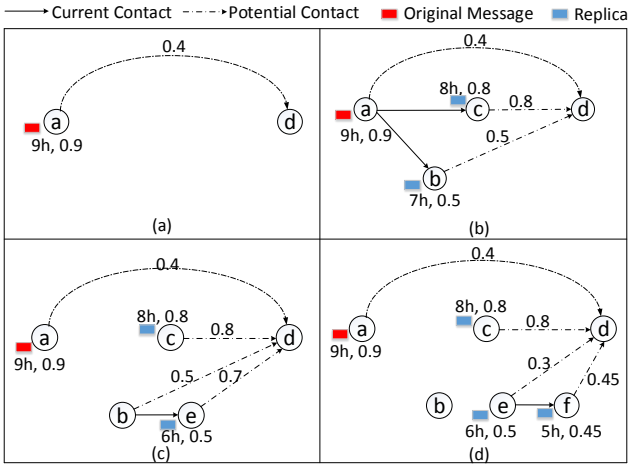

Fig. 4. Illustration of replication and forwarding

When two nodes encounters, they will first update the expected delivery rate via direct contact for each message they carry. It might happen that, for a replica carried by node a, the probability for node a to deliver the replica directly to its destination drops below the replica's expected delivery rate as time elapses. In this case, node a will resume message replication and spread more replicas to fulfill the expected delivery rate. As illustrated in Fig. 4(d), node e carries a replica with an expected delivery rate of 0.5, but the probability for node e to directly send the replica to node d drops to 0.3 after one hour. So node e spreads a new replica to node f to help fulfilling the expected delivery rate of 0.5. It can be seen that, driven by the message's expected delivery rate, message replication is performed online by taking into account the real-time network conditions and the replication history.

IV. Performance Evaluation

In this section, we evaluate the performance of our scheme through trace-driven simulations. We first investigate the impact of parameter setting on routing performance, and then compare with Epidemic, PRoPHET, and 3R in terms of average delivery rate and message overhead. We used the following two representative DTN traces:

Cambridge [11]: This trace includes Bluetooth sightings by graduate students carrying iMotes for six days, and contains more than 200 devices. In our simulations, we extend it to 2 months by duplicating the original trace, and add some noise by shifting each duplicated contact for δ seconds where δ is randomly chosen from [-4500, 4500]. So the pairwise contacts in this data trace have a high degree of repetitions.

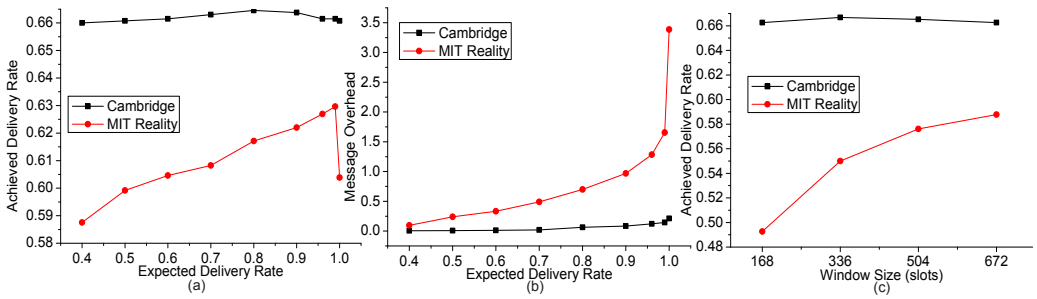

Fig. 5. (a) and (b) show the impact of the expected message delivery rate; (c) shows the impact of window size;

MIT Reality [12]: We used the *devicespan* subtrace that records Bluetooth contacts among faculties and students at MIT for ~ 9 months, and this trace contains more than 20,000 devices. The pairwise contacts in this data trace have a medium degree of repetitions, and it contains many random contacts.

A. Simulation Setup

Due to the large difference on node popularity, the selection of sources and destinations for the two traces are slightly different. In the Cambridge trace, 30 most active nodes are selected as sources and destinations, whereas in the MIT trace the destination for each message is selected randomly from 20 most active nodes during the lifetime of that message. This selection avoids negligible delivery ratio caused by numerous inactive nodes. All protocols use the same message trace, which is generated as follows: each source generates a message with a probability of 0.1 in every 600 seconds, and the message destination is randomly chosen from the set of destinations. The lifetime of each message varies from 4 hours to 1 week, and the size of a message varies from $2k$ bytes to $100k$ bytes.

To make fair comparisons, in each simulation run, the network is warmed up for one month with no messages generated, and there is also no message generated in the last week to avoid any wandering packet at the end of the simulation. All parameters in PRoPHET are configured to the recommended values [14]. Unless specially noted, the parameters in FGDR are configured as follows: the sliding window is set to 4 weeks (672 slots, 1 hour per slot), the expected delivery rate for each message is 0.9, and the minimum expected delivery rate γ_m is set to 0.1.

B. Impact of Protocol Parameter Configuration

1) Expected Message Delivery Rate: Figs. 5 (a) and (b) show the achieved average message delivery rate and the message overhead with the variation of the expected message delivery rate. For the Cambridge trace, the achieved message delivery rate slightly increases with the increase of the expected message delivery rate, and drops a little at the end. The message overhead slightly increases, and the maximum is just 0.21. This is due to the regularity of the Cambridge trace as it is produced by duplicating the original trace. Even though noise has been added, the contacts between nodes still exhibit strong weekly repeat patterns. Hence, most p_m^is in Eq. (4) are 1, and very few replicas need to be generated. Since our approach can capture the contact patterns, nodes can propagate replicas to the right relays so that the message delivery rate and overhead remain roughly constant. This demonstrates that our scheme can achieve superior performance when contacts between nodes exhibit a high degree of repeated patterns.

For the MIT trace, with the increase of the expected message delivery rate, the achieved delivery rate increases gradually except a sharp drop when the expected delivery rate is set to 1. The message overhead is proportional to the expected delivery rate when the expected delivery rate is no larger than 0.9. Beyond that, the message overhead increases significantly, and reaches around 3.5 when the expected delivery rate is 1. This phenomenon can be explained using Eq. (4). Theoretically, $P_m(k)$ can never reach 1 if all p_m^is are smaller than 1. Since the MIT trace contains many random contacts, most p_m^is are smaller than 1. So each source node continues spraying replicas upon encountering good forwarders unless the expected delivery rate has been fulfilled.

2) Size of the Sliding Window: Fig. 5 (c) shows the impact of the sliding window size on the achieved average message delivery rate. For the Cambridge trace, the size of the sliding window does not have much impact on the average delivery rate. This is due to the regularity of this trace since maintaining only one week contact history is still able to precisely represent the nodes contact patterns. For the MIT trace, the achieved average delivery rate improves with the increase of the window size. Since contacts in this trace do not have perfect repeat patterns, a larger sliding window can better represent the contact patterns and improve the accuracy of contact prediction.

C. Comparison with Existing Schemes

1) Cambridge Trace: Figs. 6 (a) and (b) show the average delivery rate and the message overhead of the four schemes by varying the size of the per-node message buffer. For Epidemic, the achieved message delivery rate increases with the increase of message buffer size as it performs blind flooding. For the other three schemes, the achieved message delivery rate first increases with the increase of buffer size and then remains steady. This is because all these three schemes control message replication, and message delivery rate cannot be further improved when the message buffer is large enough to carry messages. It can be seen that both FDGR and 3R outperform PRoPET for all buffer settings,

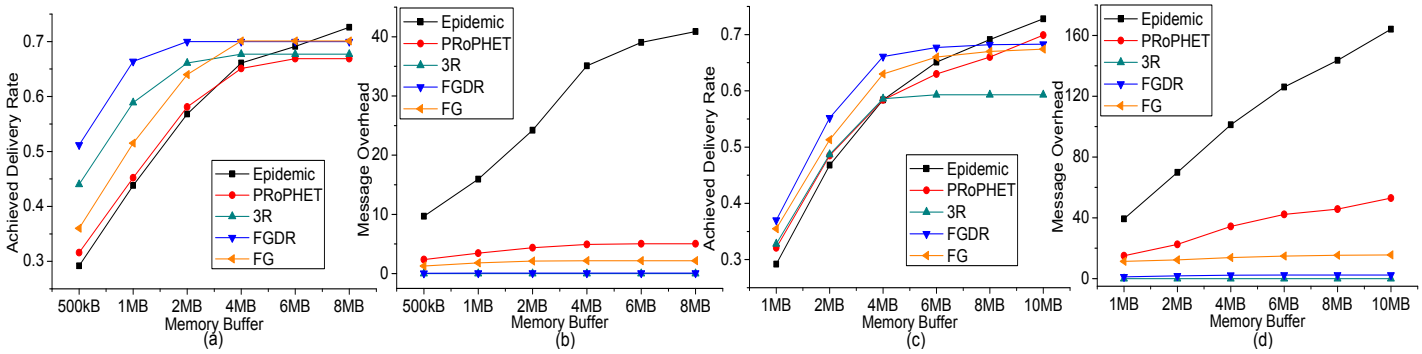

Fig. 6. Performance of FGDR, Epidemic, PRoPHET and 3R: (a) and (b) for Cambridge trace, and (c) and (d) for MIT trace

especially when message buffer is small. This is because FDGR and 3R use fine-grained contraction history to predict future contacts, by which the accuracy of contact prediction has been greatly improved. When the message buffer is set to 1M bytes/node, the average delivery rate achieved by FGDR is improved by 47% in comparison with PRoPHET. It can be seen that FGDR outperforms 3R under all buffer settings due to the delivery rate driven message replication. However, it is worth noting that the message overhead in FGDR is not significantly increased and remains very stable, as shown in Fig. 6 (b). When the message buffer is set to 6M bytes/node, the message overhead of FGDR is only 0.09, whereas PRoPHET and Epidemic have a message overhead of 5 and 39, respectively. These demonstrate that FGDR can achieve remarkable performance when nodes are memory-constrained and have regular contact patterns.

2) MIT Trace: Figs. 6 (c) and (d) show the performance of the four schemes using the MIT Reality trace. Generally speaking, the average message delivery rate and message overhead have the same trends as those in the Cambridge trace. FGDR achieves higher average delivery rate than the other schemes in most cases, but with extremely low overhead. When message buffer is set to 4M bytes/node, the average delivery rate achieved by FGDR is improved by 15.3% in comparison with PRoPHET, but the message overhead in PRoPHET is more than 38 times of that in FGDR. 3R achieves roughly the same message delivery rate as PRoPHET when message buffer is no larger than 4M bytes/node, but performs much worse than PRoPHET when message buffer is large. This is because the MIT trace contains numerous random contacts that show no discernible patterns, and 3R keeps only a single copy for each message which is not enough to achieve high message delivery rate. When message buffer is set to 10M bytes/node, Epidemic and PRoPHET performs slightly better than FGDR, but this slight improvement was achieved by a significant increase on message overhead.

V. Conclusions

In this paper, we propose an efficient routing scheme called FGDR for DTNs to achieve high message delivery rate with extremely low message overhead. In FGDR, the accuracy of contact prediction is improved by characterising contact history in a fine-grained manner, and message replication is controlled in a decentralised way by taking into account the expected message delivery rate, the replication history, and the real-time network conditions. Trace-based simulations demonstrate that FGDR can achieve remarkable performance when contacts between nodes have a high degree of repetitions, and performs better than other schemes even when the contacts between nodes are more random, especially when message buffer size is small. Future work is to investigate more efficient solutions to store the bit matrix to further reduce the storage requirement for maintaing fine-grained contact information.

References

[1] P. Hui, V. Kumar, M. Marathe, and A. Jianhua Shao ;and Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. Mobile Comp.*, vol. 11, no. 5, pp. 821 – 834, 2012.

[2] Y. Zhang and J. Zhao, "Social network analysis on data diffusion in delay tolerant networks," in *Proceedings of MobiHoc*, 2009, pp. 345–346.

[3] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, pp. 779–782, 2008.

[4] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[5] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption tolerant networks," in *Proceedings of INFOCOM*, 2006, pp. 1–11.

[6] Q. Yuan, I. Cardei, and J. Wu, "An efficient prediction-based routing in disruption-tolerant networks," *IEEE Trans. Parallel Distrib. Syst*, vol. 23, no. 1, pp. 19–31, 2012.

[7] X. Chen, J. Shen, T. Groves, and J. Wu, "Probability delegation forwarding in delay tolerant networks," in *Proceedings of ICCCN*, 2009, pp. 1–6.

[8] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Tech. Rep., 2000.

[9] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of WDTN*, 2005, pp. 252–259.

[10] S. Moon and A. Helmy, "Understanding periodicity and regularity of nodal encounters in mobile networks: A spectral analysis," in *Proceedings of GLOBECOM*, 2010, pp. 1–5.

[11] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2006-01-31)," http://crawdad.org/cambridge/haggle/, 2006.

[12] N. Eagle and A. S. Pentland, "CRAWDAD data set mit/reality (v. 2005-07-01)," Downloaded from http://crawdad.org/mit/reality/, 2005.

[13] M. Lenczner, B. Grgoire, and F. Proulx, "CRAWDAD data set ilesansfil/wifidog (v. 2007-09-07)," http://crawdad.org/ilesansfil/wifidog/, 2007.

[14] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic routing protocol for intermittently connected networks," draft-irtf-dtnrg-prophet-10, 2012.