Adaptive Message Routing and Replication in Mobile Opportunistic Networks for Connected Communities

HAIBO ZHANG, LUMING WAN, and YAWEN CHEN, University of Otago
LAURENCE T. YANG, St. Francis Xavier University
LIZHI PENG, University of Jinan

Mobile opportunistic networking is a promising technology that can supplement existing cellular and WiFi networks to provide desirable services for smart and connected communities. Message routing is the most compelling challenge in mobile opportunistic networks due to the lack of contemporaneous end-to-end paths and the resource constraints at mobile devices. To improve the probability of successful message delivery, most existing routing schemes use the past contact history to predict future contacts for message forwarding, and exploit message replication and redundancy for multicopy routing. However, most existing prediction-based routing schemes simply use the average pairwise contact probability as the routing metric and neglect the benefits of exploring fine-grained contact information such as pairwise repeated contact patterns to improve the accuracy of predicting future contacts. Moreover, there is no efficient mechanism that can adaptively control message replication in a decentralized manner to achieve both high probability of successful message delivery and low message overhead. To address these problems, we present FGAR, a routing protocol designed for mobile opportunistic networks by leveraging fine-grained contact characterization and adaptive message replication. In FGAR, contact history is characterized in a fine-grained manner with timing information using a sliding window mechanism, and future contacts are predicted based on the fine-grained contact information, thereby improving the accuracy of contact prediction. We further design an efficient message replication scheme in which message replication is controlled in a fully decentralized manner by taking into account the expected message delivery probability, the replication history, and the quality of the encountered device. A replica is generated only when it is necessary to fulfill the expected message delivery probability. We evaluate our scheme through trace-driven simulations, and the simulation results show that FGAR outperforms existing schemes. In comparison with PRoPHET, FGAR can achieve more than 20% improvement on average on successful message delivery, whereas the message overhead has been reduced by a factor up to 15.

CCS Concepts: • **Networks → Network layer protocols**; **Mobile ad hoc networks**; *Network simulations*;

Additional Key Words and Phrases: Message replication, routing, repeated contact pattern, fine-grained contact characterization, connected communities

ACM Reference format:
Haibo Zhang, Luming Wan, Yawen Chen, Laurence T. Yang, and Lizhi Peng. 2017. Adaptive Message Routing and Replication in Mobile Opportunistic Networks for Connected Communities. *ACM Trans. Internet Technol.* 18, 1, Article 2 (October 2017), 22 pages.
https://doi.org/10.1145/3122984

Authors' addresses: H. Zhang, L. Wan, and Y. Chen, Department of Computer Science, University of Otago, Dunedin, 9016, New Zealand; emails: {haibo, lwan, yawen}@cs.otago.ac.nz; L. T. Yang, Department of Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada; email: ltyang@stfx.ca; L. Peng, School of Information Science and Engineering, University of Jinan, Jinan, 250022, P.R. China; email: plz@ujn.edu.cn.

https://doi.org/10.1145/3122984

1 INTRODUCTION

The smart and connected community is an emerging area that presents a new way of designing, building, and managing communities based on information and communications technology (ICT), especially wireless and mobile communications, to achieve social, economic, and environmental sustainability. Everything in smart communities is expected to be connected and intelligent, enabling the delivery of services and information sharing from safety and security to healthcare, education, and entertainment. In 2009, Cisco launched the global initiative for Smart+Connected Communities using the network as the platform to transform physical communities to connected communities run on networked information [8]. It is highlighted that the foundation for future communities is the network and the information it carries [7]. Hence, the design and development of intelligent networking and communication technologies to weave together people, service, community assets, and information have become the key challenges in realizing smart and connected communities.

Although the past decade has witnessed substantial growth in access to ICTs, the deployment of ICT infrastructures and technologies is still lagging behind the ever-growing demands on anywhere and anytime Internet connectivity, especially for rural communities. According to the 2016 edition of the Measuring the Information Society (MIS) report [13], only around 40% of the world's population has access to the Internet. Even though the advances in cellular and WiFi networks have increased the Internet connectivity, these technologies alone are not sufficient to meet the requirements for smart and connected communities. Cellular networks can provide extensive coverage, but most mobile users are on plans with no or limited mobile data quota. Moreover, data communication through 3G/4G networks is still not cheap, especially for accessing multimedia data. Although WiFi networks can offer cheap wireless Internet access, they suffer from the drawback of limited coverage.

Mobile opportunistic networking, an extension of the mobile ad hoc network (MANET) by exploiting short-range wireless technologies such as WiFi Direct and Bluetooth for device-to-device communication, is a promising technology that supplements the existing cellular and WiFi networks to provide desirable service for smart and connected communities. A key feature of the mobile opportunistic network is that it is self-organizing without any infrastructure support, and devices communicate opportunistically upon encountering each other. When two devices come into the communication range of each other (we call it *contact* hereafter), they can share the carried data objects on the fly. In the past decade, mobile devices such as smartphones and tablets have become very popular. Almost all mobile devices have Bluetooth radios, and most of the recent smartphones support both Bluetooth and WiFi Direct for device-to-device communication. An opportunistic network composed of a large number of mobile devices together with the existing infrastructure-based cellular and WiFi networks forms a powerful network platform for connected communities and enables many application scenarios. For example, residents in remote areas can access email and Web resources by using public buses as data mules [26]. In vehicular networks, a vehicle can share traffic information upon encountering other vehicles and can seek help from other vehicles for transmitting data to roadside units when it is out of the communication range of the roadside units [3]. In earthquake disaster recovery, where most traditional communications can be interrupted, mobile opportunistic networks can be used to share rescue information between nearby smartphones [17].

Due to high mobility and short wireless transmission range, mobile opportunistic networks do not have contemporaneous end-to-end paths for message delivery, which makes routing in mobile opportunistic networks a challenging problem. Since mobile devices are carried by human beings and move around, contacts between devices in mobile opportunistic networks usually exhibit a high degree of repetition due to the daily scheduled routines of the persons who carry these devices

[11]. Hence, most existing routing schemes use the history of the previous contacts to predict the future contacts. In most prediction-based routing schemes such as PRoPHET [21], MaxProp [3], and PER [32], the contact history for each pair of devices 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 the long-term average contact probability instead of the probability that two devices will contact during the lifetime of a message. For example, let's say two devices a and b frequently contact one another during the day but not at night. Even though the encounter predictability between a and b is high, device a is not good at relaying any message that must be delivered to device b during the night. Hence, the lack of fine-grained information such as contact time will undermine the accuracy of predicting future contacts and consequently degrade the routing performance.

To enhance the success probability of message delivery, most existing routing protocols for mobile opportunistic networks are designed based on data replication and data redundancy [3, 21, 30, 33]. For instance, multiple copies (also called *message 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 [21], a message replica is generated for any encountered device that has a larger probability to contact the message's destination. In Spray and Wait [28], a fixed number of message replicas per message are generated and spread to the network. Spreading a small number of replicas may not achieve high message delivery reliability, but overspreading may also degrade routing performance due to frequent 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 to be generated and spread.

Novelty and contributions. In this article, we present FGAR, a new routing scheme for mobile opportunistic networks by exploiting the following two mechanisms: (1) *F*ine-*G*rained contact characterization and (2) *A*daptive message *R*eplication.

To the best of our knowledge, 3R [30] is the only protocol that employs fine-grained contact history to predict future contacts. However, 3R simply records the fine-grained information of each contact (e.g., contact starting and ending time) locally at each device, which is not memory efficient, especially for devices that have a high frequency for contacting others. In this article, we propose a novel sliding window mechanism to extract fine-grained information from the contact history and store the characterized contact information using a bit matrix data structure in a memory-efficient way. Based on the bit matrix structure, the probability that two devices will encounter within a given time period (e.g., the lifetime of a message) can be effectively estimated, and this probability is more accurate than the long-term average contact probability for contact prediction. Moreover, the sliding window mechanism can quickly respond to the changes on contact patterns by timely incorporation of new contact information and eliminating outdated history data for predicting future contacts.

Existing replication-based routing schemes such as PRoPHET and Spray and Wait cannot adaptively control message replications and thus cannot avoid the problem of over- or underspreading message replicas. We propose a novel decentralized mechanism to adaptively control message replication based on the expected success probability on message delivery, the replication history, and the quality of the encountered device. A replica is generated only when it is needed to fulfill the expected success probability on message delivery, thereby avoiding over- and underspreading message replicas.

We evaluated the performance of FGAR through trace-driven simulations. Simulation results show that FGAR can achieve much higher message delivery reliability than existing schemes with very low message overhead. In comparison with 3R, the percentage of successfully delivered messages has been improved by 21% on average, whereas the memory requirement has been

significantly reduced. In comparison with PRoPHET, FGAR achieves 20% improvement on successful message delivery on average, whereas the message overhead has been reduced by a factor up to 15.

The rest of this article is organized as follows. Section 2 summarizes the related work. Section 3 presents our motivations. Section 4 uses opportunistic mobile data offloading as a case study to explain the benefits of our scheme. Section 5 illustrates the representation of fine-grained contact history and contact prediction. Section 6 presents our message replication and forwarding scheme. Section 7 evaluates the performance of our scheme through trace-driven simulations. Section 8 concludes the article and discusses future work.

2 RELATED WORK

Mobile opportunistic networks have been extensively studied in recent years for many applications, such as vehicular ad hoc networking [3], mobile data offloading [12, 18], remote animal monitoring [15], disaster recovery [17], social network analysis [25, 34], and community detection [31]. Research on message forwarding in opportunistic networks originates from Epidemic routing [29], in which each message is flooded to all reachable devices in the network. Epidemic can achieve the best message delivery in scenarios with massive message buffers but does not perform well when devices have limited resources. Hence, Epidemic is commonly used as a benchmark to evaluate the performance of new routing protocols. To avoid blind flooding, controlled-flooding protocols have been proposed. PRoPHET, one of the few routing protocols that have an IETF draft [20], avoids blind flooding by pruning the Epidemic distribution tree. This pruning is done by allowing a device to send a message replica to an encountered device only when the encountered device has a larger probability than itself to contact the destination. Delegation forwarding [6] is very similar to PRoPHET except each message is replicated and forwarded based on the maximum quality observed so far instead of only the quality of the current encountered device. Spray and Wait [28] controls message overhead by limiting the number of message replicas generated in the spraying phase and then relies on direct contact to deliver each replica to the destination. The protocols proposed in [1] and [16] leverage utility values to manage the forwarding of each message and its replicas. Rather than directly optimizing the delivery probability, they focus more on minimizing the average delivery delay based on the utility value. Additionally, the utility value can be employed to drop packets when the memory allocated for buffering message is full. The OOF protocol proposed in Liu and Wu [22] simply limits the number of forwarding hops for message transmission, and a new metric taking into account both the remaining hop count and the residual time-to-live is proposed for message forwarding. With a fixed number of message copies, OOF can maximize the message delivery probability. Different from OOF, our scheme can adaptively control the number of message replicas to achieve a high percentage of successful message deliveries with low message overhead.

Recent studies show that contacts between devices in opportunistic networks, especially for community-based applications, commonly exhibit some repeating patterns. Discovering and exploiting these repeating patterns can greatly improve successful message delivery. Zhang et al. [33] studied bus-based delay-tolerant networks and observed that the intercontact time aggregated at a route level exhibit periodic behavior. Moon and Helmy [23] analyzed the periodicity in the encounter pattern by using power spectral analysis. It was observed that devices with low encounter frequency show strong periodicity. Predict and Relay (PER) [32] takes into account the contact time and location information, and exploits a time-homogeneous semi-Markov process model to represent device mobility as transitions between landmarks. Gao et al. [10] exploited transient social contact patterns for message forwarding.

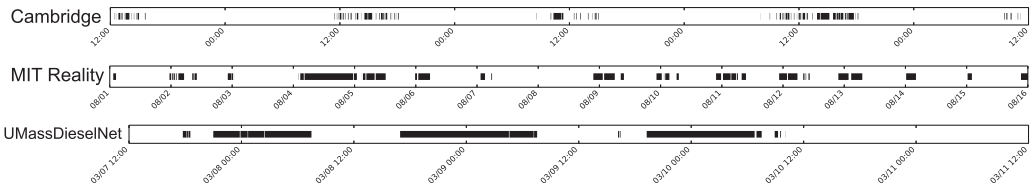

Fig. 1. Contacts between two randomly chosen devices in the Cambridge, MIT Reality, and UMassDieselNet traces.

Some studies have used the dynamic connection status for contact prediction. The schemes proposed in Vu et al. [30] and Jain et al. [14] use the information learned from each contact to compute the shortest path between pairwise devices for message routing. Chen and Shen [5] construct a fully connected distributed social map for message routing. The scheme proposed in the work of Li et al. [19] forms several communities to classify devices into different groups, and messages are propagated to the appropriate community in which destinations are located. The schemes in the work of Boldrini et al. [2], Musolesi and Mascolo [24], and Nguyen and Giordano [25] record the context information to summarize the social relationships of users. The context information includes the name, address, organization, and so on. Each attribute in the context has different degree of impact on the future contact prediction based on its associated weight. The message routing decision is made based on the likelihood of the context between pairwise devices. The higher the likelihood, the closer the message to its destination. However, it is a concern that such complicated context structure and contents can occupy too much memory for mobile devices, especially for a large population community with numerous numbers of frequently contacted devices. In addition, the lack of message replication control makes them suffer from a relative high message overhead.

To the best of our knowledge, 3R [30] is the only routing protocol using fine-grained history for contact prediction. In 3R, previous contacts are classified and maintained according to the day (weekday/weekend) and the time slot in which they occurred. Messages are forwarded based on the fine-grained contact information. However, the past contacts are stored directly at each device in a database, which is not storage efficient. Moreover, 3R uses only a single message copy and cannot achieve the best message delivery reliability.

3 MOTIVATIONS

3.1 Fine-Grained Contact Characterization

Figure 1 shows the pairwise contacts between two active devices randomly chosen from three data traces: Cambridge [27], MIT Reality [9], and UMass DieselNet [4].

It can be seen that most contacts occurred roughly in the same period of a day. This repeating pattern is useful for predicting future contacts. However, most existing schemes do not fully take this advantage. In PRoPHET, when two devices a and b contact, they update their encounter predictability, denoted by $P(a, b)$, as follows:

$$P(a, b) = P_{old}(a, b) + (1 - \delta - P_{old}(a, b)) * P_{encounter}, \tag{1}$$

where $P_{old}(a, b)$ is the encounter predictability before the current contact occurs, $P_{encounter}$ is a scaling factor at which the probability increases upon encounters, and δ is a small positive value to set an upper bound on $P(a, b)$ [20]. Forwarding messages based on this metric is simple but may not be efficient due to the inaccuracy of contact probability estimation, as illustrated by the following example. As shown in Figure 1, the two devices a and b in the Cambridge trace frequently made contact between 11 am and 6 pm on a daily basis, and $P(a, b)$ will be large (e.g., 0.9) according to

Equation (1). Suppose that device a generates a message at 8 pm and needs to send it to device b in 12 hours. Device a only knows that it has a large probability of contacting device b but is not aware that the two devices never made contact between 8 pm and 8 am in the past. Since $P(a, b)$ is high, the probability for device a to spray a replica to another encountered device is small. However, if device a has the timing information of each previous contact, it can estimate the probability that two devices will encounter in the lifetime of the message, thereby improving the accuracy of contact prediction.

In addition, the pairwise device contact patterns may change over time, and these changes must be reflected in a timely manner to improve the accuracy of future contact prediction. In PRoPHET, if devices a and b do not encounter during a time interval, $P(a, b)$ is reduced using the following aging mechanism:

$$P(a, b) = P_{old}(a, b) \ * \lambda^{K/\delta}, \tag{2}$$

where $\lambda \in [0, 1]$ is the aging constant, δ is the length of a time unit, and K is the amount of time elapsed since the last time $P(a, b)$ was aged [20]. However, the performance of the aging mechanism heavily depends on the parameters λ and δ, as an improper setting will cause either slow or fast aging. Moreover, all devices use the same setting for λ and δ, and it is hard to choose a proper setting since different devices may exhibit different behaviors.

3.2 Adaptive Message Replication

A key challenge in designing routing protocols for mobile opportunistic networks is to achieve high probability of successful message delivery with low message overhead, as devices are usually resource constrained. To illustrate this issue, we carried out trace-driven simulations to evaluate three representative schemes: Epidemic, PRoPHET, and 3R. In our simulation, we used the Cambridge data trace, which is a trace of Bluetooth sightings by graduate students carrying small devices (iMotes) for 6 days, and we duplicated the original trace to expand it to 1 month. Messages were generated with random sources and destinations. Each message has a lifetime randomly chosen from 2 hours to 1 week, and the message size varies from 2k bytes to 100k bytes The three schemes were evaluated using the same message trace and compared using the following two metrics:

(1) *Average message delivery rate*: Ratio of the number of delivered messages to the number of created messages
(2) *Average message overhead*: Ratio of the number of replicas to the number of created messages.

We carried out simulations with different settings on the message buffer size at each device. A message can be dropped due to either deadline expiration or buffer overflow. For the latter case, messages are dropped based on the first-in-first-out (FIFO) policy.

Figures 2 and 3 show the average message delivery rate and message overhead for the three schemes with different settings on message buffer size, respectively. It can be seen that for unlimited buffer size, Epidemic achieves the highest average message delivery rate (0.725) with extremely high message overhead. The average message overhead is approximately 41, and 98.2% of the created messages and replicas were dropped. PRoPHET achieves an average message delivery rate of 0.669 with a significant drop on the number of replicated messages, but the message overhead still remains high (approximately six replicas were generated per created message). It is worth noting that 3R achieves an average message delivery rate of 0.67 with no message overhead. Even though 3R uses a single message copy, the average message delivery rate is not dropped too much, as it uses fine-grained encounter information for contact prediction.

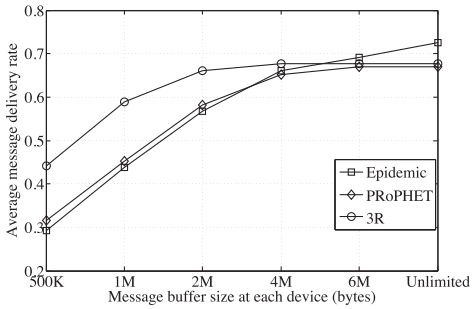

Fig. 2. Average message delivery rate versus message buffer size/device.

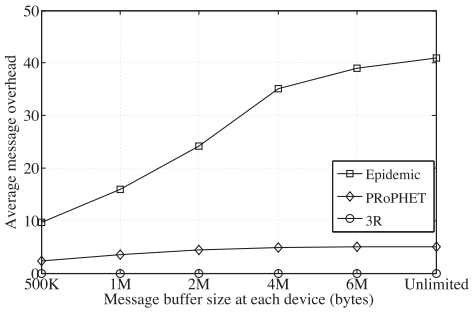

Fig. 3. Average message overhead versus message buffer size/device.

It is also worth noting that for the scenarios with a message buffer size no larger than 2M bytes per device, the average delivery rate achieved by Epidemic is even lower than PRoPHET and 3R. This is because a large portion of the message replicas have to be dropped due to message buffer overflow. Hence, message overhead has a big impact on the performance of a routing scheme in resource-constrained mobile opportunistic networks. A single copy per message is not sufficient to achieve a high message delivery rate, but the pruning mechanism in PRoPHET is too simple and not efficient.

From Figures 2 and 3, it can be seen that a large portion of the replicated messages are not helpful for improving the message delivery rate. Our motivation is to avoid the generation of the unnecessary message replicas to achieve a high message delivery rate with very low message overhead. Different from existing works, we aim to develop an efficient message replication scheme in which each device can adaptively control the replication of the messages it carries. A replica is generated only when it can contribute to the improvement of successful message delivery, thereby avoiding over- and underspreading message replicas.

4 A CASE STUDY: OPPORTUNISTIC MOBILE DATA OFFLOADING

As discussed in previous sections, the message routing scheme developed in this article has many application scenarios. In this section, we use opportunistic mobile data offloading as a case study and explain how the proposed routing scheme can be used to deliver data originally targeted for cellular networks using complementary short-range communication technologies such as Bluetooth and WiFi.

With the increasing popularity of mobile devices such as smartphones and tablets, the past few years have witnessed a tremendous growth not only in the number of mobile users but also in the demand for mobile Internet access. The explosive growth in mobile traffic, especially multimedia data, will make the cellular networks overloaded and congested due to the limited network capacity. Mobile data offloading is a promising technology to release the burden of cellular networks by using complementary network technologies for delivering data originally targeted for cellular networks. There are two types of offloading: on-the-spot and delayed [18]. In on-the-spot offloading, mobile devices simply switch to WiFi once WiFi connectivity is available, and most smartphones already perform on-the-spot offloading by default. In delayed offloading, each data transfer is associated with a deadline, and the device that requests the data transfer will first try other networks. If the data transfer is not completed after the deadline, it will come back to use the cellular network to complete the data transfer. As illustrated by the example given in Figure 4, Bob wants to send 10 pictures to his friend John, but they do not have to be available to John immediately. He opts to

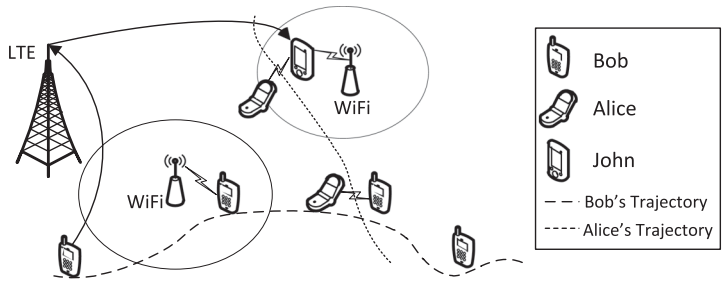

Fig. 4. Message replication and forwarding.

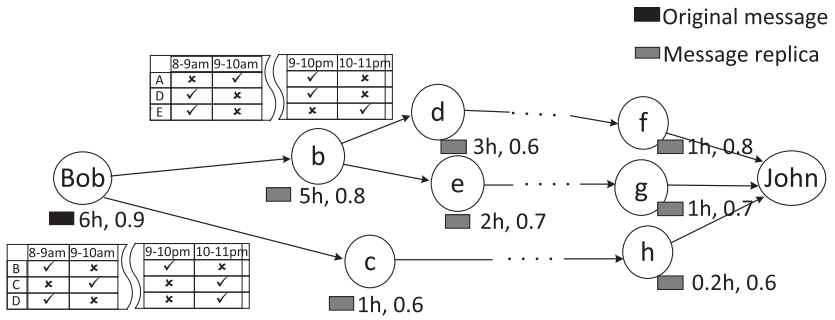

Fig. 5. Message replication and forwarding.

have them delivered within 6 hours if this will avoid any carrier charge. He first sends a text message via cellular networks to inform John that he will send him 10 pictures in the following 6 hours and asks John to confirm receipt of the pictures after the deadline. After 1 hour, Bob encounters John's colleague Alice and forwards a copy of the pictures to her, and hopes she will encounter John in the following 5 hours to forward the pictures to John. After 3 hours, Bob has a chance to connect to WiFi and also sends a copy of the pictures through WiFi to John. It is possible that John received all pictures since he spent most of the time in his office with a connection to the Internet, or that John received most pictures from Alice but one picture was not successfully transferred due to the short contact. John informs Bob about the missing picture, and Bob resends it through the cellular networks. Since data transfer in delay offloading heavily depends on the opportunistic contacts between mobile devices or mobile devices and WiFi access points, delay offloading is also called *opportunistic mobile data offloading*. A large portion of mobile data traffic, such as weather forecasts, electronic newspapers, and movie trailers, does not have strict real-time constraints and thereby is suitable to be offloaded to other networks. Many studies have been carried out to perform efficient opportunistic offloading to maximize the amount of offloaded data [12, 19].

The routing scheme developed in this article is for end-to-end message delivery and can be used for opportunistic mobile data offloading. Each message is characterized by a source address, a destination address, a delivery deadline, and an expected success probability of message delivery imposed by either the user or the upper-layer application or set with a default value. For example, a value of 1 indicates that the message is expected to be reliably delivered to the destination within its deadline. Each device uses a time-slotted sliding window to maintain the most recent contact history, as shown in Figure 5. 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 device keeps track of the devices it contacted in each time slot. In this way, the encounter probability

between any two devices in any time period can be estimated based on this fine-grained contact history. As shown in Figure 5, if Bob decides to send the pictures to John at 8 am, the probability for Bob to directly contact device a in the following 6 hours can be estimated using only the past contacts occurring between 8 am and 2 pm, thereby improving the accuracy of contact prediction.

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

As illustrated in Figure 5, it is assumed that Bob expects to send the pictures to John in 6 hours with an expected success probability of 0.9. After 1 hour, Bob encounters device a and learns that device a has a probability of 0.8 to contact John. As the expected success probability of the original message has not been fulfilled, Bob sends a replica to device a. The expected success probability for this replica is set to 0.8, and the deadline is set to 5 hours from now. After 2 hours, Bob encounters device b and sends another replica to device b. Since then, the expected success probability (0.9) has been fulfilled, and Bob stops spreading more replicas to other encountered devices. As the pairwise contact probabilities may change over time, each device needs to dynamically control message replication. For example, device a did not encounter John in the following 3 hours and generates another two replicas for devices c and d to help fulfill the expected success probability of 0.8. Thus, each device adaptively controls the generation of message replicas and route messages in a multihop manner, and there is no constraint on the number of hops for a message to reach the destination.

5 CONTACT CHARACTERIZATION AND PREDICTION

5.1 Contact Detection and Representation

Each device needs to maintain the fine-grained information of past contacts, such as the address of the encountered device and the contact time. However, directly recording this information is not efficient in terms of both storage and retrieval speed, as most devices are resource constrained, and a device may encounter hundreds of devices or even more. In our scheme, we use the bit matrix data structure to characterize the contacts occurring in the sliding window. Let L_w denote the length of the time-slotted sliding window, and let L_s represent the length of each time slot in the window. Each device a maintains a bit matrix, denoted by $\mathcal{M}_a$, to represent the recent contacts occurring between device a and its encountered devices during the time period covered by the sliding window. $\mathcal{M}_a$ is defined as

$$\mathcal{M}_a[b][k] = \begin{cases} 1, \text{ if } \max_{i=1}^{n_k} l_{ab}^k(i) \geq \alpha \text{ or } \sum_{i=1}^{n_k} l_{ab}^k(i) \geq \beta, \\ 0, \text{ otherwise,} \end{cases} \tag{3}$$

where n_k is the number of contacts occurring 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 or the accumulative contact time in time slot k is no smaller than a threshold β. The reason for 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

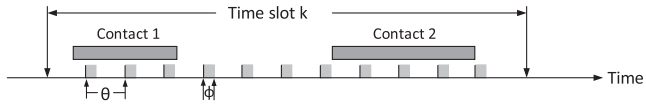

Fig. 6. Illustration of the estimation of $l_{ab}^{k}(i)$.

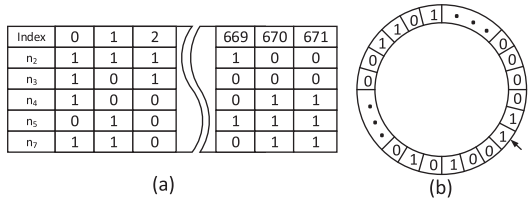

Fig. 7. (a) Bit matrix with 672 slots. (b) Circular buffer for one row.

two devices to transfer all messages that need to be exchanged. However, the two devices contact several times in slot k, and the aggregated contact time is long enough to exchange messages. This eliminates instantaneous contacts that are not helpful in message forwarding, as counting these instantaneous contacts will lead to incorrect message forwarding decisions.

In mobile opportunistic networks, the contacts between devices are commonly detected through periodic scanning. Several studies have been performed on energy-efficient probing in mobile opportunistic networks [35]. We assume that each device scans to discover neighbors every θ seconds (e.g., 60 seconds), and each scanning lasts for ϕ seconds (e.g., 10 seconds). Then the duration of each contact (i.e., $l_{ab}^{k}(i)$ in Equation (3)) can be estimated based on this periodic scanning. As shown in Figure 6, a contact may span multiple scanning periods. Since a contact may start and end at any time in the $(\theta - \phi)$ seconds before the first time and after the last time it was detected, we estimate the duration of the i^{th} contact in the k^{th} slot as follows:

$$l_{ab}^{k}(i) = rand(0, \theta - \phi) + n_i \times \theta + rand(0, \theta - \phi), \tag{4}$$

where n_i is the number of consecutive scans in which b is discovered by a, and the two $rand(0, \theta - \phi)$s represent the amount of time the contact lasts before the first time and after the last time it was detected, respectively.

Figure 7(a) gives an example of the bit matrix used to represent a sliding window with 672 slots (i.e., $L_w = 4$ weeks and $L_s = 1$ hour). As shown in Figure 7(a), each row represents the fine-grained contact information for one encountered device within the sliding window. Each row is implemented using a circular buffer data structure with size of L_w/L_s bits, since it is well suited for FIFO buffering without the need for element shuffling for insertion operations. Moreover, all circular buffers at one device can share the same write pointer, as each column in the bit matrix is updated together slot by slot. Thus, each device periodically scans, calculates the maximum and accumulated contact time in the current slot, moves the writer pointer to the current slot, and then updates the corresponding column in the bit matrix according to Equation (3). Additionally, data in the circular buffers can be fast retrieved based on the matrix indexes. The storage requirement based on bit matrix will be discussed in Section 7.3in comparison with other schemes.

5.2 Contact Prediction

Contacts between different devices may exhibit different patterns. For example, postgraduates may have weekly meetings with supervisors, friends may have a weekly social gathering, and

employees in the same department may frequently contact one another during workdays. The sliding window mechanism and the bit matrix representation allow us to capture these patterns.

In the slotted sliding window, the history for 1 week is represented with $\frac{24\times7}{L_s}$ slots, then the number of weeks that the window maintains for the contact history is as follows:

$$n_w = L_w \Big/ \frac{24\times 7}{L_s} = \frac{L_w \times L_s}{168}. \tag{5}$$

We assume that 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 in its circular buffer 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 a positive remainder. Let $p_{ab}(k)$ denote the probability that devices a and b will contact in slot k. Then $p_{ab}(k)$ can be estimated using the history characterized by these n_w slots as follows:

$$p_{ab}(k) = \frac{\sum_{m=0}^{n_w-1} w_m \times \mathcal{M}_a[b][index(m)]}{n_w}, \tag{6}$$

where w_m is the weight assigned to slot $index(m)$. Different slots can have different weights. For example, more recent slots can have larger weights to timely reflect changes on contact patterns.

Suppose that device a carries a message/replica that needs to be delivered to device b during the time period $[t_s, t_e]$. Since t_s and t_e may appear at any time in a slot, the slot in which t_s or t_e resides may not be fully covered by $[t_s, t_e]$. Let k be the slot in which t_s resides. The number of slots that $[t_s, t_e]$ covers, including the incomplete slots, can be computed as

$$n_s = \begin{cases} 1, & t_e - t_s + \eta_k \le L_s, \\ 1 + \lceil \frac{(t_e - t_s) - (L_s - \eta_k)}{L_s} \rceil, & \text{otherwise}, \end{cases} \tag{7}$$

where η_k is the amount of time from the start of slot k until t_s, which can be easily obtained since each device needs to maintain the start time of the current slot to update the bit matrix. If $t_e - t_s + \eta_k \le L_s$, t_s and t_e are in the same slot.

Let $p_{ab}(t_s, t_e)$ be the probability that device a can send the message directly to device b during $[t_s, t_e]$. According to Equation (6), the probability for two devices to contact in a slot is estimated based on only the past contacts with contact time long enough for message transfer. Hence, the message delivery fails only when there is no such contact during the period of $[t_s, t_e]$. We assume that t_s and t_e of a contact occur in a slot with uniform distribution, and let k be the slot in which t_s resides. Thus, $p_{ab}(t_s, t_e)$ can be estimated as follows:

$$p_{ab}(t_s, t_e) = \begin{cases} p_{ab}(k) \times \frac{t_e - t_s}{L_s}, & t_e - t_s + \eta_k \le L_s, \\ 1 - \prod_{i=k}^{k+n_s-1} (1 - \bar{p}_{ab}(i)), & \text{otherwise}. \end{cases} \tag{8}$$

When t_s and t_e are in different slots, slot k is the first slot and slot $k + n_s - 1$ is the last slot, and

$$\bar{p}_{ab}(i) = \begin{cases} p_{ab}(k) \times \frac{L_s - \eta_k}{L_s}, & i = k; \\ p_{ab}(i), & i \in (k, k + n_s - 1); \\ p_{ab}(k + n_s - 1) \times \frac{\eta_{(k+n_s-1)}}{L_s} & i = k + n_s - 1. \end{cases} \tag{9}$$

It can be seen that the distribution of t_s and t_e is used only for estimating the contact probability in the incomplete slots. Thus, the distribution will not significantly influence the accuracy of the estimation for $p_{ab}(t_s, t_e)$ even if it is not uniform.

ALGORITHM 1: Replication and Forwarding at Sender

```
/* Upon encountering device b:                                          */
for each message m in a's buffer do
    if Dest_m == b then
        Forward message m to device b;
        Delete message m;
    else
        Add < ID_m, Dest_m, T_m, γ_m > to SV_a;
Send SV_a to device 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;
```

6 MESSAGE REPLICATION AND ROUTING

6.1 Message Replication and Forwarding

Each message or replica m has a lifetime T_m and an expected success probability E_m, where E_m is the accumulative success probability 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 device 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 the devices in $\mathbb{F}_m = \{f_m^1, f_m^2, \ldots, f_m^k\}$, and the expected success probabilities to be fulfilled by these k replicas and their descendants are given in $\mathbb{P}_m = \{E_m^1, E_m^2, \ldots, E_m^k\}$. The success probability that is expected to be achieved after generating these k replicas, denoted by $P_m(k)$, can be computed as follows:

$$P_m(k) = 1 - \prod_{i=0}^{k} (1 - E_m^i), \tag{10}$$

where $E_m^0 = p_m^0$ is the probability of delivering m from the device 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 Equation (10) 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 device that carries m will stop generating more replicas.

Algorithms 1 and 2 give the pseudocode for message replication and forwarding at sender (device a) and receiver (device b), respectively. When device a encounters device b, device a first delivers all of its messages that are destined to device b. After that, device a checks how device b can help deliver the other messages it carries. To do this, device 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 success probability for a replica of m that is used to eliminate bad forwarders. After generating SV_a, device a sends it to device b. Upon receiving SV_a, for each

ALGORITHM 2: Replication and Forwarding at Receiver

```
/* Upon receiving SV_a from device a:                                    */
for each < ID_m, Dest_m, T_m, γ_m > in SV_a do
    if device b carries a replica with ID_m then
        Continue;
    else
        Calculate p̄_{bDest_m} according to Equation (8).
        if p̄_{bDest_m} ≥ γ_m then
            Add < ID_m, p̄_{bDest_m} > to the ACK message;
Send ACK to device a;
```

$< ID_m, Dest_m, T_m, \gamma_m >$, device b will check if it has already carried a replica with ID_m. If not, device 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 Equation (8). If $\bar{p}_{bDest_m} \geq \gamma_m$, it indicates that device b is a good forwarder for m, and $< ID_m, \bar{p}_{bDest_m} >$ is added to the ACK message. In the end, device b sends the ACK message to device a and informs which messages it chooses to help deliver. After receiving the ACK message, for each $< ID_m, \bar{p}_m >$, device a decides whether to forward message m or send a replica of m to device b based on the following rules:

Rule1: If $P_m(k) < E_m$ & $\bar{p}_m \geq E_m$, device a directly forwards the message/replica m to device b. This is because the expected success probability allocated for m has not been fulfilled, but device b can achieve an expected success probability no smaller than E_m if it carries m. Hence, there is no need to generate a new replica.

Rule2: If $P_m(k) < E_m$ & $\bar{p}_m < E_m$, device a generates a new replica and sends it to device b. The expected success probability for the new replica is the probability for device b to contact the destination, and device a updates $P_m(k)$.

Rule3: If $P_m(k) \geq E_m$ & $\bar{p}_m > p_m^0$, device a forwards m to device b. This is because even though the expected success probability E_m has been fulfilled, device b can provide higher delivery probability than device a.

Illustrating example. As shown in Figure 8(a), device a has a message to be delivered to device d in 9 hours with an expected success probability of 0.9. The probability for device a to send the message to device d through direct contact is 0.4. As shown in Figure 8(b), device a encounters device c after 1 hour and forwards a replica to device c according to Rule2. The expected success probability for this new replica is set to 0.8. After 2 hours, device a encounters device b and forwards it another replica. After that, the expected success probability for the original message at device a has been fulfilled, and device a stops spreading replicas. As shown in Figure 8(c), device b carries a replica and encounters device e, which has a larger delivery probability (0.7) than b. According to Rule3, device b forwards the carried replica to device e.

When two devices encounter, they will first update the expected success probability via direct contact for each message they carry. It might happen that for a replica carried by device a, the probability for device a to deliver the replica directly to its destination drops below the replica's expected success probability as time elapses. In this case, device a will resume message replication and spread more replicas to fulfill the expected success probability. As illustrated in Figure 8(d), device e carries a replica with an expected success probability of 0.5. but the probability for device e to directly send the replica to device d drops to 0.3 after 1 hour. Thus,

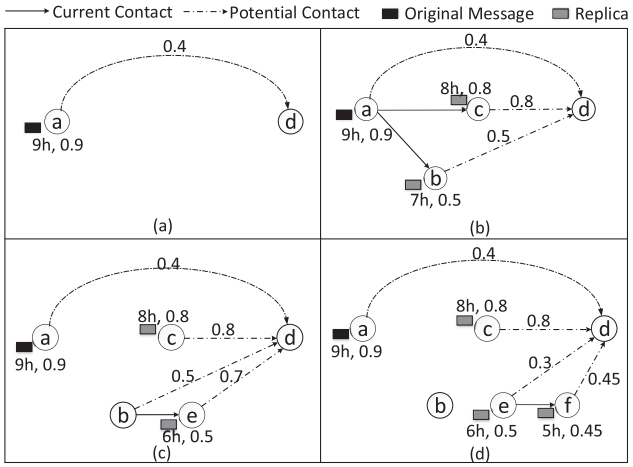

Fig. 8. Illustration of replication and forwarding.

device e spreads a new replica to device f to help fulfill the expected success probability of 0.5.

Even though the example given in Figure 8 only shows message routing for two or three hops, there is no limitation on the length of the message routing chain. According to Rule1, a message or replica can be forwarded to any encountered device that can fulfill the expected success probability. According to Rule3, a message or replica can be forwarded to any encountered device that has a larger delivery probability than the current device with no limitation on hop count. The generation of message replicas is adaptively controlled using Rule2 by taking into account the real-time network conditions and the replication history.

6.2 Automatic Adjustment of Relay Selection Threshold γ_m

In our scheme, each message replica m is associated with a parameter γ_m for forwarder selection. A large γ_m prevents the sending of replicas to encountered devices with low message delivery probability, which reduces message overhead but may not spread enough replicas to fulfill the expected success probability. However, a small γ_m will lead to the generation of many replicas to encountered devices with low message delivery probability, which will increase the message overhead. Since each device in our scheme maintains only local information, it cannot decide whether another device is a good forwarder for a certain message until it encounters that device. Additionally, a device cannot predict whether the current encountered device is the best or if it will encounter a better one later. Thus, we use the following scheme to dynamically adjust the message forwarding threshold:

$$\gamma_m^{new} = \min\left\{\gamma_m^{old} \times \frac{\Delta_m^t}{T_m} \times \frac{E_m}{\Delta_m^e}, \gamma_0\right\}, \tag{11}$$

where Δ_m^t is the remaining lifetime for replica m, Δ_m^e is the success probability that is expected to be fulfilled by the already generated replicas, and γ_0 is an upper bound for γ_m. It can be seen that the smaller the Δ_m^t, the smaller the γ_m^{new} so that more replicas will be generated and spread to more forwarders to fulfill the expected success probability. In the same way, the larger the Δ_m^e, the larger the γ_m^{new} to ensure that only good forwarders are selected to reduce message overhead.

7 PERFORMANCE EVALUATION

In this section, we evaluate the performance of FGAR through trace-driven simulations. We first investigate the impact of parameter setting on routing performance and then compare it to Epidemic, PRoPHET, 3R, and Spray and Wait in terms of average delivery rate, message overhead, and memory requirement. We used the following three representative traces:

Cambridge [27]: This trace includes Bluetooth sightings by graduate students carrying iMotes for 6 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 $[-4500, 4500]$. Thus, the pairwise contacts in this data trace have a high degree of repetitions.

MIT Reality [9]: We used the *devicespan* subtrace that records Bluetooth contacts among faculties and students at MIT for approximately 9 months; 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.

UMass DieselNet [4]: The trace contains bus-to-bus communication during the spring semester of 2006. The trace contains 36 buses in total. Some of the pairwise contacts appear with a high degree of repetitions but still contain random contacts and noise.

7.1 Simulation Setup

Due to the large difference in device popularity, the selection of sources and destinations for the three traces are slightly different. In the Cambridge trace, the 30 most active devices are selected as sources and destinations, whereas in the MIT trace, the destination for each message is selected randomly from 20 most active devices during the lifetime of that message. There is no limitation on selecting source and destination devices in the DieselNet trace since it contains only 36 devices, and most of them have a relative high contact frequency. This selection avoids a negligible delivery ratio caused by numerous inactive devices. We use the same message trace to compare the performance of different protocols, and the message trace is generated as follows: each source generates a message with a probability of 0.1 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 1 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 [20]. Unless specially noted, the parameters in FGAR are configured as follows: the sliding window is set to 4 weeks (672 slots, 1 hour per slot), the expected success probability for each message is 0.9, and λ_0 in Equation (11) is set to 1. For Spray and Wait, we use its *binary* version and run simulations with different settings on the maximum number of replicas generated per message. We observe that Spray and Wait achieves the highest average message delivery rate with the following settings and use these settings for comparison: 5 replicas per message for Cambridge, 70 replicas per message for MIT Reality, and 10 replicas per message for UMass DieselNet trace.

7.2 Impact of Protocol Parameter Configuration

7.2.1 Expected Message Success Probability. Message replication in FGAR is driven by the expected success probability. The higher the expected success probability, the more replicas may be generated to fulfill the expected delivery rate. Figure 9(a) and (b) show the achieved average message delivery rate and the message overhead with the variation of the expected success probability.

Table 1. Storage Requirement for Contact Prediction

Protocol	Cambridge		MIT	
	Max (bytes)	Avg (bytes)	Max (bytes)	Avg (bytes)
PRoPHET	2,230	119	99,970	420
3R	339,300	1,677	665,730	1,844
FGAR (672, 1)	7,830	387	42,480	74
FGAR (336, 2)	4,176	206	22,656	40
FGAR (168, 3)	2,349	116	12,744	23
FGAR (84, 4)	1,436	71	7,788	18

For the Cambridge trace, the achieved message delivery rate slightly increases with the increase of the expected success probability 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 devices still exhibit strong weekly repeat patterns. Hence, most p_m^is in Equation (10) are 1, and very few replicas need to be generated. Since our approach can capture the contact patterns, devices 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 devices exhibit a high degree of repeated patterns.

For the MIT trace, with the increase of the expected success probability, the achieved delivery rate increases gradually except a sharp drop when the expected success probability is set to 1. The message overhead is proportional to the expected success probability when it is no larger than 0.9. Beyond that, the message overhead increases significantly and reaches around 3.5 when the expected success probability is 1. This phenomenon can be explained using Equation (10). 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. Thus, each source device continues spraying replicas upon encountering good forwarders unless the expected success probability has been fulfilled.

7.2.2 Size of the Sliding Window. Figure 9(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 1 week of contact history is still able to precisely represent the device's 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.

7.2.3 Weight on Contact History. In Equation (6), different slots in the sliding window can have different weights in computing the contact probability. Figure 9(d) shows the achieved average message delivery rate using different weighting schemes. For the Cambridge trace, there is no significant difference for the two weighting schemes due to the regularity of this data trace. But for the MIT Reality trace, a higher message delivery rate is achieved by giving larger weight to more recent slots, as this can timely reflect changes on contact patterns.

7.3 Storage Requirement for Contact Prediction

Table 1 compares the storage requirement for contact prediction in different schemes in terms of the maximum storage required for the busiest device and the average storage consumption for

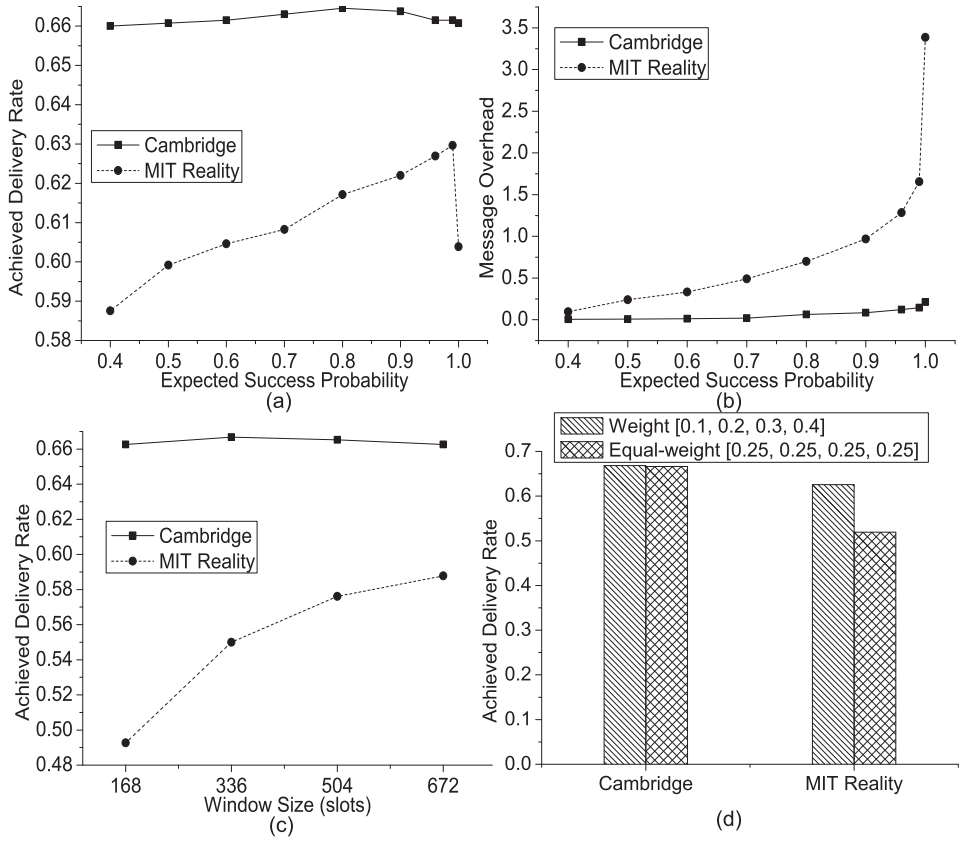

Fig. 9. The impact of the expected message delivery rate (a, b). The impact of window size (c). The impact of weighting (d).

the entire population. Each device uses its MAC address (6 bytes) as its identity. In PRoPHET, each device needs to record the encounter probability (float type, 4 bytes) for each encountered device. In FGAR, each device locally records the bit matrix, whereas in 3R, each device needs to record the date and time for each contact. It can be seen from Table 1 that 3R has the largest storage requirement, as it needs to store the information for each contact, and FGAR does not consume extremely large storage in comparison with PRoPHET even though it maintains fine-grained contact information. In addition, FGAR is able to consume less memory by characterizing the history into a sparser sliding window, but the performance is just slightly degraded, as discussed in Section 7.2.2. For the Cambridge trace, when $L_w = 672$ and $L_s = 1$, both the maximum and average storage requirement for FGAR is around three times as that for PRoPHET. For the case where $L_w = 168$ and $L_s = 3$, FGAR consumes roughly the same memory as PRoPHET. For the MIT trace, both the maximum and average storage requirement of FGAR is much less than PRoPHET at all stages. This is because the MIT trace contains a large number of inactive devices. Although PRoPHET only needs to record the device address and its contact probability, a device has to keep the entire list from the encountered devices, even though the records contain lots of the information for the devices it has never seen. However, a device using FGAR only needs to maintain the history for its encountered devices, and also the history is dumped if the previously encountered devices disappear for a long time. Due to a huge difference in the population of two traces, the number of history records on

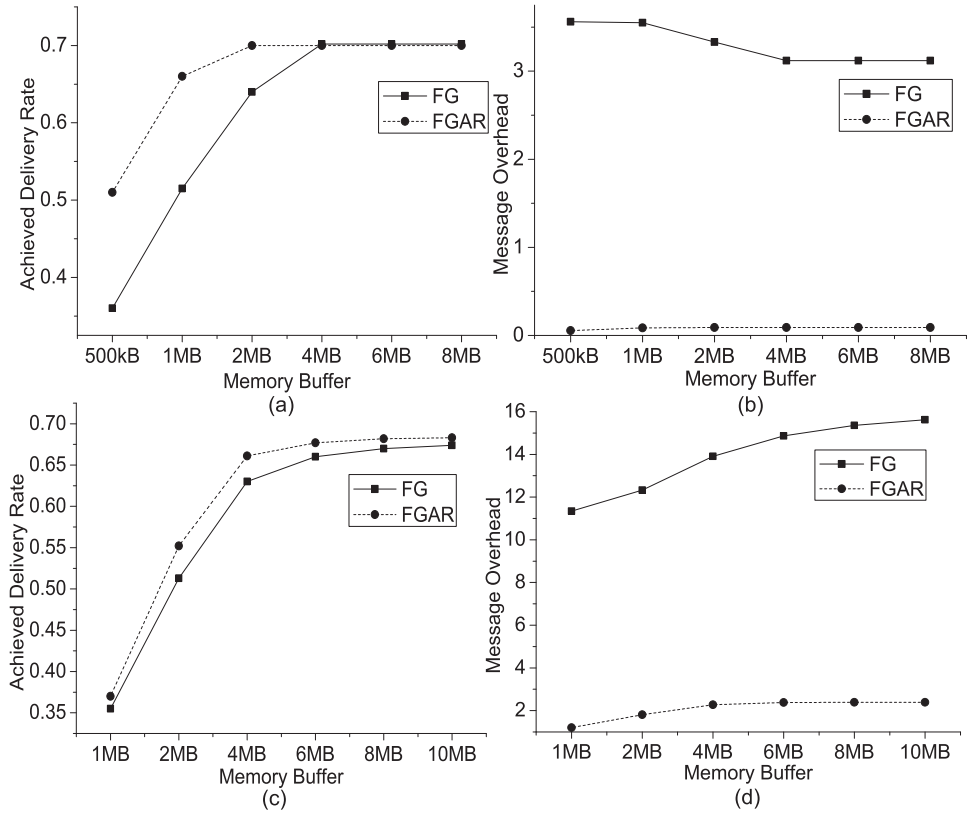

Fig. 10. Performance of FG and FGAR: Cambridge trace (a, b) and MIT trace (c, d).

average is 87 per device when using PRoPHET and 4.3 for FGAR in the Cambridge trace, whereas it is 42.02 for PRoPHET and 0.82 for FGAR using the MIT trace.

7.4 Comparison with Existing Schemes

In this section, we first show the advantages of dynamic message replication control. Figure 10 compares the performance improvement achieved by fine-grained contact characterization and adaptive message replication. FG is a simplified version of FGAR by removing its replication control process. In the Cambridge trace, it can be seen that FGAR outperforms FG on delivery rate when the memory buffer is small, and then they have the same performance when the buffer is large enough. However, FG generates 24.6 times more message replicas than FGAR with a sufficient memory buffer. Due to the lack of message replication control, the redundant message replicas consume too much memory, which leads to the discarding of other packets due to memory overflow. The lost packets will consequently result in the degradation of routing performance. In the MIT trace, the performance gap is much smaller than that in the Cambridge trace. FGAR and FG have a very close average message delivery rate for all memory settings, but FGAR generates only 1.7 replicas/message on average, whereas FG requires at least 11 replicas. This indicates that contact prediction based on fine-grained history is reliable, and an intellectual message replication control can remove unnecessary message copies, thus further enhancing the routing performance.

Figure 11 shows the comparison of our scheme with Epidemic, PRoPHET, Spray and Wait, and 3R using the Cambridge, MIT, and DieselNet traces.

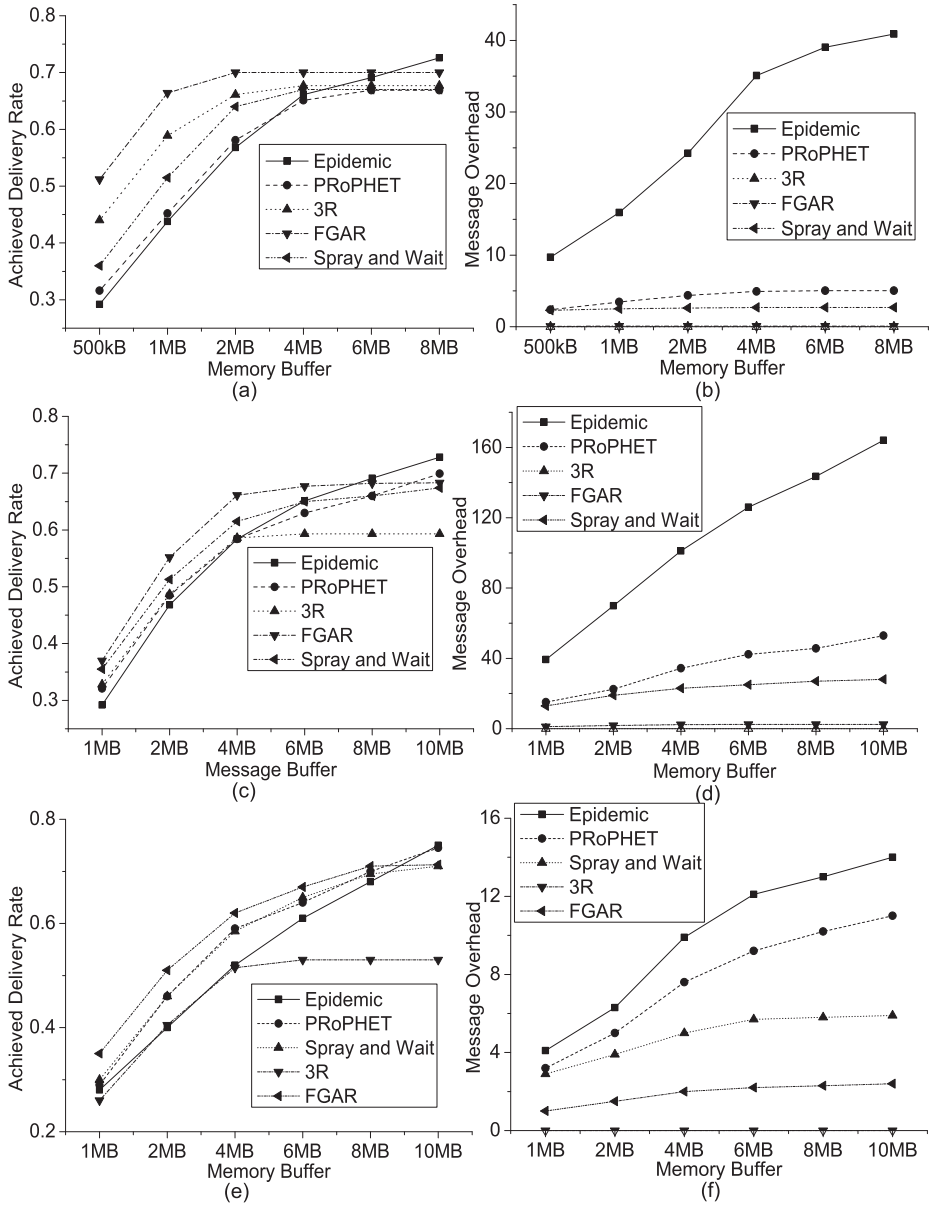

Fig. 11. Performance of FGAR, Epidemic, PRoPHET, and 3R: Cambridge trace (a, b), MIT trace (c, d), and DieselNet trace (e, f).

Cambridge Ttrace. Figure 11(a) and (b) show the average delivery rate and message overhead of the five schemes by varying the size of the per-device 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 four schemes, the achieved message delivery rate first increases with the increase of buffer size and then remains steady. This is because all four schemes control message replication, and the message delivery rate cannot be further improved when the message buffer is large enough to carry messages. It can be seen that both FGAR and 3R outperform PRoPHET

and Spray and Wait for all buffer settings, especially when the message buffer is small. This is because FGAR 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/device, the average delivery rate achieved by FGAR is improved by 47% in comparison with PRoPHET. It can be seen that FGAR outperforms 3R under all buffer settings due to the delivery rate–driven message replication. However, it is worth noting that the message overhead in FGAR is not significantly increased and remains very stable, as shown in Figure 11(b). When the message buffer is set to 6M bytes/device, the message overhead of FGAR is only 0.09, whereas Spray and Wait, PRoPHET, and Epidemic have a message overhead of 2.9, 5, and 39, respectively. This demonstrates that FGAR can achieve remarkable performance when devices are memory constrained and have regular contact patterns.

MIT trace. Figure 11(c) and (d) show the performance of the five 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. FGAR achieves a higher average delivery rate than the other schemes in most cases but with extremely low overhead. When the message buffer is set to 4M bytes/device, the average delivery rate achieved by FGAR is improved by 15.3% in comparison with PRoPHET, but the message overhead in PRoPHET is more than 38 times that in FGAR. 3R achieves roughly the same message delivery rate as PRoPHET when the message buffer is no larger than 4M bytes/device but performs much worse than PRoPHET when the 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 a high message delivery rate. When the message buffer is set to 10M bytes/device, Epidemic and PRoPHET perform slightly better than FGAR, but this slight improvement was achieved by a significant increase in message overhead. FGAR outperforms Spray and Wait at all memory cases, whereas Spray and Wait generates at least 20 times more message replicas than FGAR.

UMass DieselNet trace. Figure 11(e) and (f) show the performance of the five schemes using the DieselNet trace. It can be seen that the overall trends of both achieved delivery rate and message overhead have the same appearance as in the previous two traces. FGAR still gains a higher achieved delivery rate than the other four protocols when devices have a small message buffer. When the memory buffer is set to 2M bytes/device, the delivery rate of FGAR is increased 9.8% over PRoPHET and 10.1% better in comparison with Spray and Wait. The message overhead for FGAR at this stage is only 1.72, whereas it is 5.12 for PRoPHET and 3.99 for Spray and Wait. This is because FGAR takes advantage of fine-grained historical information characterization for its accurate prediction, and the adaptive message replication can intellectually eliminate unnecessary message replicas. Due to the limitation on the number of message replicas in Spray and Wait, it always has a lower message overhead than PRoPHET. As the memory buffer increases, Epidemic and PRoPHET take advantage of flooding. Hence, it slightly outperforms FGAR. However, Epidemic generates 14.65 replicas per message on average, and PRoPHET has 11.8 for this, whereas it is only 3.37 for FGAR.

8 CONCLUSIONS

Mobile opportunistic networking is a promising technology that supplements the existing cellular and WiFi networks to provide desirable service for smart and connected communities. To address the challenge of routing messages in mobile opportunistic networks, we propose an efficient routing scheme called *FGAR* that can achieve a high message delivery rate with extremely low message overhead. In FGAR, the accuracy of contact prediction is improved by characterizing contact history in a fine-grained manner, and message replication is controlled in a decentralized way by

taking into account the expected message delivery rate, the replication history, and the real-time network conditions. Trace-based simulations demonstrate that FGAR can achieve remarkable performance when contacts between devices have a high degree of repetitions, and it performs better than other schemes even when the contacts between devices are more random, especially when the message buffer size is small. Future work will investigate more efficient solutions to store the bit matrix to further reduce the storage requirement for maintaining fine-grained contact information.

REFERENCES

[1] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. 2007. DTN routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review* 37, 4, 373–384.

[2] Chiara Boldrini, Marco Conti, Jacopo Jacopini, and Andrea Passarella. 2007. HiBOp: A history based routing protocol for opportunistic networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM'07)*. 1–12.

[3] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. 2006. MaxProp: Routing for vehicle-based disruption tolerant networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'06)*. 1–11.

[4] John Burgess, John Zahorjan, Ratul Mahajan, Brian Neil Levine, Aruna Balasubramanian, Arun Venkataramani, Yun Zhou, Bruce Croft, Nilanjan Banerjee, Mark Corner, and Don Towsley. 2008. The Umass/Diesel Dataset (v. 2008-09-14). Retrieved September 19, 2017, from http://crawdad.org/umass/diesel/20080914

[5] Kang Chen and Haiying Shen. 2014. SMART: Utilizing distributed social map for lightweight routing in delay-tolerant networks. *IEEE/ACM Transactions on Networking* 22, 5, 1545–1558.

[6] X. Chen, J. Shen, T. Groves, and J. Wu. 2009. Probability delegation forwarding in delay tolerant networks. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN'09)*. 1–6.

[7] Cisco. 2010. Smart+Connected Communities: Changing a City, a Country, the World. Retrieved September 19, 2017, from http://www.cisco.com/c/dam/en_us/solutions/industries/docs/scc/09CS2326_SCC_BrochureForWest_r3_112409.pdf.

[8] Cisco. 2017. Smart+Connected Communities. Retrieved September 19, 2017, from http://www.cisco.com/c/en/us/solutions/industries/smart-connected-communities.html.

[9] Nathan Eagle and Alex (Sandy) Pentland. 2005. The MIT/Reality Dataset (v. 2005-07-01). Retrieved September 19, 2017, from http://crawdad.org/mit/reality/.

[10] W. Gao, G. Cao, T. La Porta, and J. Han. 2013. On exploiting transient social contact patterns for data forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing* 12, 1, 151–165.

[11] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *Nature* 453, 779–782.

[12] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan. 2012. Mobile data offloading through opportunistic communications and social participation. *IEEE Transactions on Mobile Computing* 11, 5, 821–834.

[13] ITU. 2016. Measuring the Information Society Report. Retrieved September 19, 2017, from http://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2016/MISR2016-w4.pdf.

[14] Sushant Jain, Kevin Fall, and Rabin Patra. 2004. Routing in a delay tolerant network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04)*. 145–158.

[15] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. 2002. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM SIGARCH Computer Architecture News* 30, 5, 96–107.

[16] Amir Krifa, Chadi Barakat, and Thrasyvoulos Spyropoulos. 2008. An optimal joint scheduling and drop policy for delay tolerant networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM'08)*. 1–6.

[17] Silvia Krug, Peggy Begerow, Atheer Al Rubaye, Sebastian Schellenberg, and Jochen Seitz. 2014. A realistic underlay concept for delay tolerant networks in disaster scenarios. In *Proceedings of the International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'14)*. 163–170.

[18] Kyunghan Lee, Joohyun Lee, Yung Yi, Injong Rhee, and Song Chong. 2010. Mobile data offloading: How much can WiFi deliver? In *Proceedings of the 6th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'10)*. 26:1–26:12.

[19] Zhong Li, Cheng Wang, Siqian Yang, Changjun Jiang, and Ivan Stojmenovic. 2014. Improving data forwarding in mobile social networks with infrastructure support: A space-crossing community approach. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'14)*. 1941–1949.

[20] A. Lindgren, A. Doria, E. Davies, and S. Grasic. 2012. Probabilistic Routing Protocol for Intermittently Connected Networks. Retrieved September 19, 2017, from https://tools.ietf.org/html/draft-irtf-dtnrg-prophet-10.
[21] Anders Lindgren, Avri Doria, and Olov Schelén. 2003. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review* 7, 3, 19–20.
[22] Cong Liu and Jie Wu. 2012. On multicopy opportunistic forwarding protocols in nondeterministic delay tolerant networks. *IEEE Transactions on Parallel and Distributed Systems* 23, 6, 1121–1128.
[23] S. Moon and A. Helmy. 2010. Understanding periodicity and regularity of nodal encounters in mobile networks: A spectral analysis. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'10)*. 1–5.
[24] Mirco Musolesi and Cecilia Mascolo. 2009. CAR: Context-aware adaptive routing for delay-tolerant mobile networks. *IEEE Transactions on Mobile Computing* 8, 2, 246–260.
[25] Hoang Anh Nguyen and Silvia Giordano. 2012. Context information prediction for social-based routing in opportunistic networks. *Ad Hoc Networks* 10, 8, 1557–1569.
[26] Alex (Sandy) Pentland, Richard Fletcher, and Amir Hasson. 2004. DakNet: Rethinking connectivity in developing nations. *Computer* 37, 1, 78–83.
[27] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. 2006. The Cambridge/Haggle Dataset (v. 2006-01-31). Retrieved September 19, 2017, from http://crawdad.org/cambridge/haggle/20060131/.
[28] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. 2005. Spray and Wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the ACM SIGCOMM Workshop on Delay Tolerant Networks (WDTN'05)*. 252–259.
[29] Amin Vahdat and David Becker. 2000. *Epidemic Routing for Partially-Connected Ad Hoc Networks*. Technical Report. Department of Computer Science, Duke University.
[30] L. Vu, Q. Do, and K. Nahrstedt. 2011. 3R: Fine grained encounter-based routing in delay tolerant networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM'11)*. 1–6.
[31] Eiko Yoneki, Pan Hui, and Jon Crowcroft. 2007. Visualizing community detection in opportunistic networks. In *Proceedings of the 2nd ACM Workshop on Challenged Networks (CHANTS'07)*. 93–96.
[32] Q. Yuan, I. Cardei, and J. Wu. 2012. An efficient prediction-based routing in disruption-tolerant networks. *IEEE Transactions on Parallel and Distributed Systems* 23, 1, 19–31.
[33] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang. 2007. Study of a bus-based disruption tolerant network: Mobility modeling and impact on routing. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom'07)*. 195–206.
[34] Yang Zhang and Jing Zhao. 2009. Social network analysis on data diffusion in delay tolerant networks. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'09)*. 345–346.
[35] Huan Zhou, Huanyang Zheng, Jie Wu, and Jiming Chen. 2013. Energy-efficient contact probing in opportunistic mobile networks. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN'13)*. 4629–4642.

Received April 2016; revised January 2017; accepted June 2017