Computer Networks xxx (2015) xxx–xxx

ELSEVIER

Contents lists available at ScienceDirect

Computer Networks

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

Optimal storage allocation on throwboxes in Mobile Social Networks

Q1 Bo Fan [a], Supeng Leng [a,*], Kun Yang [b], Yan Zhang [c]

[a] School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[b] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, United Kingdom
[c] Simula Research Laboratory, Fornebu 1364, Norway

A R T I C L E I N F O

Article history:
Received 20 February 2015
Revised 26 June 2015
Accepted 25 August 2015
Available online xxx

Keywords:
Mobile Social Networks
Throwbox
Deployment
Storage allocation
Data delivery

A B S T R A C T

In the context of Mobile Social Networks (MSNs), a type of wireless storage device called throwbox has emerged as a promising way to improve the efficiency of data delivery. Recent studies focus on the deployment of throwboxes to maximize data delivery opportunities. However, as a storage device, the storage usage of throwboxes has seldom been addressed by existing work. In this paper, the storage allocation of throwboxes is studied as two specific problems: (1) if throwboxes are fixed at particular places, how to allocate storage to the throwboxes; and (2) if throwboxes are deployable, how to conduct storage allocation in combination with throwbox deployment. Two optimization models are proposed to calculate the optimal storage allocation with a knowledge of the contact history of users. Real trace based simulations demonstrate that the proposed scheme is able to not only decrease data loss on throwboxes but also improve the efficiency of data delivery.

1. Introduction

Mobile Social Networks (MSNs) [1] are composed of mobile users that carry portable devices such as cellphones. As the links among users and the network topology are unstable, MSN can be regarded as a special type of Delay Tolerant Network (DTN) [2], which makes data delivery a challenging issue in MSN. Comparing with traditional path-building based routing approaches such as AODV [3] and DSR [4], *Store-carry-and-forward* strategy based schemes [5–8] are more efficient for data delivery. In these methods, mobile users can act as mobile relays and store data until the next hop is available. Such a strategy may partly overcome the intermittent links of MSN. However, these opportunistic encounter based schemes still have low delivery efficiency.

Many recent studies [9–12] focus on the utilization of *throwboxes* [13] in data delivery. Throwboxes are a type of storage devices equipped at particular places acting as stationary relays. As shown in Fig. 1, with the aid of a throwbox, data can be successfully delivered even if the two users do not encounter each other. In [14], the authors apply throwboxes in the Epidemic Routing protocol [15] and the Two-hop Multicopy Routing protocol [16]. The delivery delay and the resource consumption of the two protocols are both decreased.

Throwboxes are widely studied in recent researches. Some studies investigate throwbox deployment [13,17]. In [17], the social graph among specific locations and mobile users is explored to establish the placement of throwboxes. The work in [13] studies the combination of throwbox deployment and routing to achieve high throughput. Several throwbox-based relay strategies are proposed in [12]. In addition, the work in [18,19] propose an energy-efficiency scheme of throwboxes, in which a hardware and software architecture is proposed. However, as a storage

* Corresponding author. Tel.: 86-28-61830520.
E-mail addresses: bofanuestc@gmail.com (B. Fan), spleng@uestc.edu.cn (S. Leng), kunyang@essex.ac.uk (K. Yang), yanzhang@simula.no (Y. Zhang).

http://dx.doi.org/10.1016/j.comnet.2015.08.015
1389-1286/© 2015 Published by Elsevier B.V.

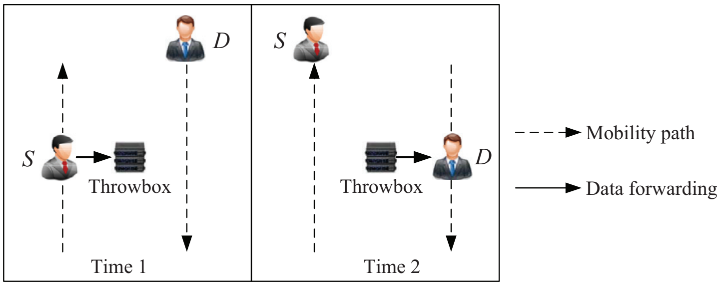


Fig. 1. User S and user D pass a throwbox at different times. User S sends a data to the throwbox firstly. Then, user D can receive the data from the throwbox.

device, the storage usage of throwboxes has been seldom studied.

In this paper, we study the optimal storage allocation of throwboxes. Since the deployment of throwboxes directly determines the usage efficiency of storage, the storage allocation problem can be discussed in the following two specific cases.

(1) *Throwboxes are fixed at particular places:* In this case, storage allocation is conducted individually on the fixed throwboxes.
(2) *Throwboxes are deployable:* Throwboxes are not deployed or can be redeployed. In this case, storage allocation can be conducted in combination with throwbox deployment.

The potential places for deploying throwboxes and storage are called user *Gathering Points* (*GPs*) [10] where a large number of users usually gather. Contact history between users and GPs is explored as a priori knowledge for estimating the storage requirement of each GP, as well as the contact strength between users and GPs. In order to calculate the optimal storage allocation, we propose a Linear Programming (LP) model for the case with fixed throwboxes and a joint optimization model for the case with deployable throwboxes.

To the best of our knowledge, this is the first work to address the optimal storage allocation on throwboxes in combination with throwbox deployment. Comparing with the existing work, the main contributions of this paper can be summarized as follows.

(1) We propose a method to evaluate the contact strength between a mobile user and a place, which fully utilizes the characters of the contacts between the user and the place, including frequency, durations and intervals.
(2) The optimal storage allocation is studied in combination with throwbox deployment. When throwboxes are deployable, both throwbox deployment and storage allocation can be solved using the proposed joint scheme.
(3) A balance between the number of throwboxes and the size of storage is achieved, so that network operators can prepare these two kinds of resources properly and avoid resource wastage.

The remainder of this paper is organized as follows. Section 2 provides a review of related researches on throwboxes. The system model of this paper is presented in Section 3, followed by the estimation of contact strength between users and GPs in Section 4. Section 5 presents the detail of storage allocation. Simulations of the proposed scheme are presented in Section 6. Finally, Section 7 concludes the paper.

2. Related works

The concept of throwbox is first introduced in [13], which defines a throwbox as a stationary relay with limited storage and power. This work addresses throwbox deployment in combination with routing designing. With different levels of knowledge, three throwbox deployment schemes are proposed. For each scheme, three different relay strategies are designed to achieve high throughput. Another work addressing throwbox deployment is [17], where the social graph among specific locations and users is exploited to determine the placement of throwboxes. Multiple metrics, such as betweenness centrality and degree centrality, are used to evaluate the importance of each potential place. Based on different metrics, several deployment schemes are presented. These two studies make excellent contributions to throwbox deployment. Nevertheless, as they both ignore storage allocation in the deployment, effective storage allocation schemes can be hardly realized with these deployment schemes, because the place selected for throwbox deployment may be not proper for storage allocation. Work [18,19] investigate an energy-efficiency scheme of throwboxes, in which a hardware and software architecture is proposed to improve the energy efficiency of throwboxes. However, as a storage device, the storage usage of throwboxes is usually ignored by existing studies.

Throwboxes are widely applied in data delivery methods. Ibrahim et al. [14] add throwboxes into two existing routing protocols, the Epidemic Routing protocol [15] and the Two-hop Multicopy Routing protocol [16] to study the enhancement of performance by using throwboxes. Simulation results show that the data delivery delay and the resource consumption of the two methods are both significantly decreased. In [12], several routing schemes are designed based on throwboxes. The authors classify nodes as source node, destination node, mobile relays and throwboxes and design five relay strategies. These strategies differ from each other only in the restriction of data forwarding among specific types of nodes. In the context of MSN, throwboxes are mainly utilized as a relay at some locations with large social popularity, such as GPs [9–11]. As these places usually

Table 1
Notation definition.

Notations and definitions		Notations and definitions	
m	The number of GPs	n	The number of users
X	Total size of storage	Y	Total number of throwboxes
$\mathbf{x}$	Storage allocation vector	$\mathbf{y}$	Throwbox deployment vector
γ	Number of real visits	λ	Contact strength
$HRoS_j$	Hard RoS of GP g_j	$SRoS_j$	Soft RoS of GP g_j
RoS_j	RoS of GP g_j	α	Weight of $HRoS_j$ in RoS_j

have a large number of visiting users, a throwbox storing data there can significantly improve the performance of data delivery. However, in the existing work, the authors simply assume each place to support a throwbox for data storing. How much storage should be allocated to each throwbox is never considered. In this case, we address this issue to fill the research gap.

3. System model

We consider a network that consists of n users $U = \{u_1, u_2, \ldots, u_n\}$ and m GPs $G = \{g_1, g_2, \ldots, g_m\}$. Users communicate with each other and with throwboxes using the short range radio of the devices, such as Wi-Fi direct and Bluetooth. The total number of throwboxes and the total size of storage are Y and X, respectively. Vector $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$ indicates the number of throwboxes deployed at each GP, and $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ denotes the size of storage allocated to the throwbox of each GP. Each GP can equip at most one throwbox, so that $y_i \in \{0, 1\}$. Data can be stored at a throwbox for a constant time T_l, which is called the storing lifecycle of data. The main notations used in this paper are listed in Table 1.

As shown in Fig. 2(a), if the throwboxes are fixed at particular places (i.e., $\mathbf{y}$ is established), the only task is to allocate storage to the throwboxes. This is a common demand in real-life situations, because the storage of throwboxes usually needs to be reallocated to adapt the varying visiting habits of users. For example, a library usually has much more visiting users at the end of a semester and needs more storage than other time. In this case, the optimal storage allocation $\mathbf{x}$ should be recalculated according to the current visit pattern of users. On the other hand, if the throwboxes are deployable (i.e., $\mathbf{y}$ is a variable) as shown in Fig. 2(b), storage allocation can be conducted in combination with throwbox deployment. In this case, the optimal throwbox deployment $\mathbf{y}$ and storage allocation $\mathbf{x}$ can be calculated jointly. The case with deployable throwboxes has been simply addressed as a preliminary work in [20] with limited simulations and discussion. This paper extends the work and fully addresses the storage allocation problem by considering both cases. Moreover, new real trace based simulations are conducted to evaluate the performance of each case as well as the comparison between them.

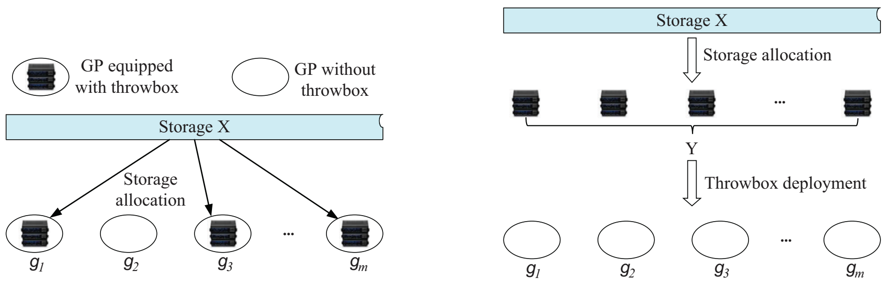
Comparing to the duration of a visit, the time cost in receiving data from a throwbox can be ignored. Most data are stored on a throwbox at the beginning of the visits. Accordingly, we can simply assume that data storing happens only once during a visit (i.e., at the beginning of it).

4. Contact strength

Contact strength denotes the strength of contact between two nodes. In this paper, we evaluate the contact strength between a user and a GP from the perspective of data receiving, which means *how possible the user can receive data from the GP*. Contact history of the user, which contains the detail of the past visits to the GP, is exploited as a priori knowledge. Such history is easy to be obtained via some information collection techniques [21].

Some researchers have studied contact strength among nodes using contact characters such as frequency [17], durations [22] or intervals [23]. However, as they all employ only one of these characters, the evaluation of contact strength may be inaccurate. For example, as shown in Fig. 3, if only frequency is considered, user A and user B should have the same contact strength with the GP. However, user A wins out because of his larger visit durations. Due to the same reason, user A defeats user C, although they have the same average interval time. User D has the same visit duration as user A. However, user A still wins out. This is because user A is able to receive the data stored during the intervals at the next visit. While, user D can only receive the data stored during the visits.

The above comparison indicates that no contact character is able to evaluate the contact strength between a user and a GP individually. Hence, we employ all the characters. Before



(a) Storage allocation on fixed throwboxes

(b) Storage allocation and throwbox deployment

Fig. 2. Storage allocation under two cases: (a) storage allocation on fixed throwboxes, and (b) storage allocation and throwbox deployment.

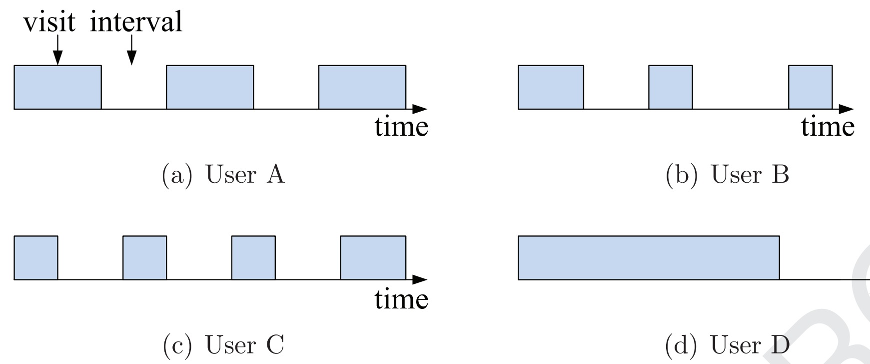


Fig. 3. Contact history of four users with the same GP: (a) user A, (b) user B, (c) user C and (d) user D.

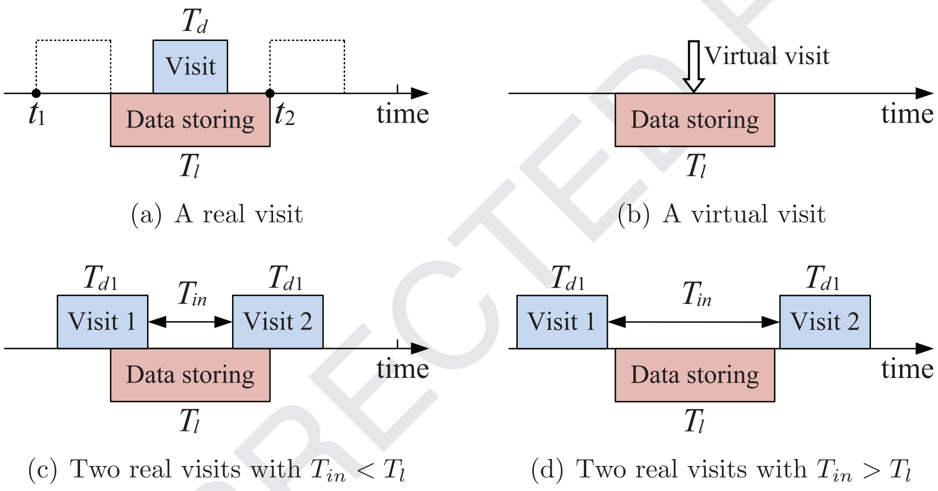


Fig. 4. Case study: (a) a real visit, (b) a virtual visit, (c) two real visits with $T_{in} < T_l$ and (d) two real visits with $T_{in} > T_l$.

introducing the estimation of contact strength, we first define two terminologies.

Definition 1. (Real visit): Visits that really occur between a user and a GP with a non-zero durations, as shown in Fig. 4(a).

Definition 2. (Virtual visit): Fictitious and instantaneous visits with zero duration, as shown in Fig. 4(b).

Unless otherwise specified, "visit" denotes real visit.

According to the contact history, a user may visit a GP for several times with different durations and intervals. It is difficult to exploit all these characters directly. Instead, we first normalize these various-length visits by converting them into virtual visits, according to the durations and intervals of the visits. Then, we employ the frequency of the virtual visits as the contact strength between the user and the GP. The conversion from real visits to virtual visits should keep the following principle: *through the virtual visits, the user should have the same chance to receive data from the GP as through the original real visits*. For each user-GP pair, we illustrate the conversion via three cases using Fig. 4.

4.1. A single real visit

Firstly, we study how to convert a single real visit into virtual visits with Fig. 4(a), in which a block indicates a period of time. For simplification, only one piece of data is considered, which is stored at the GP for time T_l. The duration of the single visit is T_d. As shown in Fig. 4(a), if block "Visit" is located between the two dashed blocks, namely if the user reaches the GP between t_1 and t_2, block "Visit" can overlap with block "Data storing" and the user can receive the data. So, the *feasible period* for data receiving is $t_2 - t_1 = T_d + T_l$.

On the other hand, as shown in Fig. 4(b), through a virtual visit, the user can receive the data only if the virtual visit occurs during T_l. Hence, the feasible period for data receiving is T_l. In order to achieve a feasible period $T_d + T_l$, $\frac{T_d+T_l}{T_l}$ virtual visits are needed. Consequently, we have the following corollary.

Corollary 1. *A real visit with duration T_d can be converted into $\frac{T_d+T_l}{T_l}$ virtual visits.*

4.2. Two adjacent real visits

Secondly, we study how to convert two adjacent real visits into virtual visits. T_{d1} and T_{d2} denote the duration of the two

Please cite this article as: B. Fan et al., Optimal storage allocation on throwboxes in Mobile Social Networks, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.08.015

visits and T_{in} denotes the interval between them. If $T_l > T_{in}$, as shown in Fig. 4(c), even though the user is absent from the GP during T_{in}, he can still receive the data stored during T_{in}, just like that he has never left during T_{in}. In other words, the two visits and the interval can be regarded as a visit from the perspective of data receiving. Therefore, we have Corollary 2.

Corollary 2. *For two adjacent real visits with interval T_{in} and durations T_{d1} and T_{d2}, if $T_l > T_{in}$, then the two visits and the interval can be regarded as a real visit with a duration $T_d = T_{d1} + T_{in} + T_{d2}$.*

Corollary 2 indicates that several close short visits can contribute the same as a long visit. Such a property is usually neglected by existing studies.

On the other hand, if $T_l < T_{in}$, as shown in Fig. 4(d), Corollary 2 is not valid. Instead, they can be converted into $\frac{T_{d1}+T_{d2}+2T_l}{T_l}$ virtual visits according to Corollary 1.

4.3. Arbitrary number of real visits

Finally, we consider an arbitrary number of real visits. Based on the above two cases, the conversion can be easily conducted. Through Corollary 2, all the adjacent real visits with $T_{in} < T_l$ can be combined into real visits. Then, based on Corollary 1, each real visit can be converted into a specific number of virtual visits.

Based on the above discussion, the contact strength of each user–GP pair can be estimates through the following steps.

(1) Select two adjacent real visits with an interval $T_{in} < T_l$ and combine them into a real visit according to Corollary 2.
(2) Repeat Step (1) until all the adjacent visits have intervals $T_{in} > T_l$.
(3) Convert each visit into virtual visits according to Corollary 1.
(4) Define the frequency of virtual visits as the contact strength λ of the user–GP pair.

5. Optimal storage allocation

Storage allocation is basically a supply–demand problem. In this section, we first analyze the *Requirement of Storage* (*RoS*) of each GP. Then, based on the analysis, we calculate the optimal storage allocation.

5.1. Demand analysis

The RoS of a GP is determined by the number of visits, the average number of data stored during each visit and the average size of stored data. It should be noted that the visits considered in this section are the real visits rather than virtual visits. This is because that data storing happens only once during a real visit. The length of a visit makes no difference to data storing. For a GP g_j, we define its *Hard Requirement of Storage* (*HRoS*) within a period of time *T* as

$$HRoS_j = \gamma_j \cdot \overline{D_j} \cdot \overline{S_j} \tag{1}$$

where γ_j is the number of visits to g_j within *T*. $\overline{D_j}$ is the average number of data stored at g_j during a visit and $\overline{S_j}$ is the average size of the data stored at g_j. HRoS is the RoS under the worst situation where the γ_j visits occur during the same T_l and cost storage simultaneously. The storage needed in such a situation is the largest. Generally, visits may occur at different time and storage can be recycled.

In the best situation where the visits are distributed uniformly in *T*, only $\frac{\gamma_j T_l}{T}$ visits happen during a T_l and the storage need is the smallest. The RoS of a GP in such a situation is called *Soft Requirement of Storage* (*SRoS*), which is given as

$$SRoS_j = \frac{\gamma_j \cdot \overline{D_j} \cdot \overline{S_j} \cdot T_l}{T}. \tag{2}$$

HRoS and SRoS are RoS under the worst situation and the best situation, respectively. If we consider a general situation, a tradeoff should be made between them. Hence, we define the RoS of g_j as

$$RoS_j = \alpha HRoS_j + (1-\alpha) SRoS_j, \quad \alpha \in [0, 1]. \tag{3}$$

α allows for adjusting the relative importance of RHoS and SHoS. The variables in RoS_j can be derived as follows:

$$\gamma_j = \sum_i^n \gamma_{ij} \tag{4}$$

where γ_{ij} is the number of real visits between u_i and g_j within *T*, namely the number of times u_i visits g_j within time *T*.

$\overline{D_j}$ and $\overline{S_j}$ can be calculated with the statistics collected during building the contact history. $\overline{D_j}$ can be calculated as

$$\overline{D_j} = \frac{\sum_k^{\gamma_j} D_{kj}}{\gamma_j} \tag{5}$$

where D_{kj} indicates the number of data stored at g_j during the *k*th visit. Let $N_j = \sum_k^{\gamma_j} D_{kj}$, then

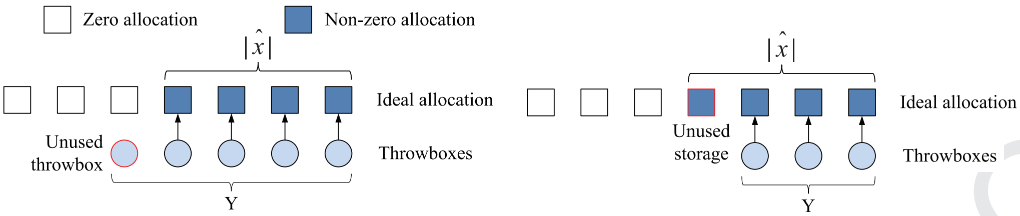
$$\overline{S_j} = \frac{\sum_l^{N_j} S_{lj}}{N_j} \tag{6}$$

where S_{lj} is the size of the *l*th data stored at g_j.

5.2. Storage allocation on fixed throwboxes

If throwboxes are fixed at particular places, storage allocation on the throwboxes is conducted individually. The objective of storage allocation is to enhance the efficiency of data delivery in the network, including improving data delivery ratio, decreasing data delivery delay and so forth. Basically, these goals can be achieved by maximizing the data delivery probability at the GPs, which is determined by the number of data stored there and the sum of contact strength between the GP and all the users. With a given storage x_j, a GP g_j can store at most $x_j/\overline{S_j}$ data, where $\overline{S_j}$ is the average size of a data. The contact strength between g_j and all the users is λ_j, where $\lambda_j = \sum_{i=0}^n \lambda_{ij}$ and λ_{ij} is the contact strength between g_j and user u_i. Hence, the data delivery probability P_j at g_j satisfies

$$P_j \propto \frac{\lambda_j \cdot x_j}{\overline{S_j}}. \tag{7}$$



(a) $Y \geq |\hat{\mathbf{x}}|$ (b) $Y < |\hat{\mathbf{x}}|$

Fig. 5. Throwbox deployment and storage allocation under the greedy method: (a) $Y \geq |\hat{\mathbf{x}}|$ and (b) $Y < |\hat{\mathbf{x}}|$.

In order to maximize the data delivery probability of the whole network, we calculate the optimal storage allocation as follows.

$$\text{Maximize} \quad F(\mathbf{x}) = \sum_{j=1}^{m} \frac{\lambda_j \cdot x_j \cdot y_j}{\overline{S_j}}$$
$$\text{subject to:} \quad \sum_{j=1}^{m} x_j = X, \ 0 \leq x_j \leq RoS_j. \tag{8}$$

Here, x_j is the size of storage allocated to GP g_j. It has a upper bound RoS_j in order to avoid wastage of storage. y_j is the number of throwboxes deployed at GP g_j. Since $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$ is already established, formula (8) is a Linear Programming (LP) problem [24], which can be easily solved with small computation cost. Such a storage allocation scheme is optimal under the established throwbox deployment scheme $\mathbf{y}$. However, it is also constrained by the deployment scheme $\mathbf{y}$. If throwboxes are deployable, storage allocation and throwbox deployment can be both conducted to achieve a better scheme.

5.3. *Joint storage allocation and throwbox deployment*

If throwboxes are deployable, $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$ is a variable. Similar as formula (8), the optimal storage allocation and throwbox deployment scheme can be given as

$$\text{Maximize} \quad F(\mathbf{x}) = \sum_{j=1}^{m} \frac{\lambda_j \cdot x_j \cdot y_j}{\overline{S_j}}$$
$$\text{subject to:} \quad \sum_{j=1}^{m} x_j = X, \ 0 \leq x_j \leq RoS_j,$$
$$\sum_{j=1}^{m} y_j = Y, \ y_j \in \{0, 1\}. \tag{9}$$

Such a joint optimization problem is NP-Hard [13]. It is too computationally expensive to solve it optimally. Consequently, we develop a two-step greedy method to solve it. Firstly, an *ideal allocation* result that ignores the restriction in the number of throwboxes is calculated. Then, considering the specific number of throwboxes, the joint scheme is established.

5.3.1. *Ideal allocation*

Suppose that Y is large enough to satisfy all the GPs, so that if $x_j > 0$, then $y_j = 1$. Formula (9) can be modified as

$$\text{Maximize} \quad F(\mathbf{x}) = \sum_{j=1}^{m} \frac{\lambda_j \cdot x_j}{\overline{S_j}}$$
$$\text{subject to:} \quad \sum_{j=1}^{m} x_j = X, \ 0 \leq x_j \leq RoS_j. \tag{10}$$

The model becomes an LP problem. As the coefficients $\lambda_j/\overline{S_j}$ are all positive, the optimal solution $\hat{\mathbf{x}} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m\}$ can be easily achieved by successively matching the RoS of each GP in the descending order of $\lambda_j/\overline{S_j}$. In other word, storage is first allocated to the GP with the largest $\lambda_j/\overline{S_j}$ to match its RoS. Then, storage is allocated to the GP with the second largest $\lambda_j/\overline{S_j}$ and so forth, until no storage or GP is left. As a consequence, all the non-zero elements $\hat{x}_j$ in $\hat{\mathbf{x}}$ satisfy $\hat{x}_j = RoS_j$.

5.3.2. *Joint scheme*

Now we consider that the number of throwboxes is constrained by Y. Let $|\hat{\mathbf{x}}|$ denotes the number of non-zero elements of $\hat{\mathbf{x}}$. As shown in Fig. 5(a), if $Y \geq |\hat{\mathbf{x}}|$, each GP that has a non-zero storage allocation in $\hat{\mathbf{x}}$ can be equipped with a throwbox. In this case, for each GP g_j, if $\hat{x}_j > 0$, then $y_j = 1$ and $x_j = \hat{x}_j$. In other words, the ideal allocation can be realized. On the contrary, if $Y < |\hat{\mathbf{x}}|$ (Fig. 5(b)), only Y GPs can be equipped with a throwbox. In order to maximize $\sum_{j=1}^{m} \lambda_j \cdot x_j \cdot y_j / \overline{S_j}$, the Y throwboxes are placed at the Y GPs that have the largest $\lambda_j \cdot \hat{x}_j / \overline{S_j}$. Then, storage are allocated to these throwboxes according to the ideal solution $\hat{\mathbf{x}}$. Namely, for each of the y selected GPs, $y_j = 1$ and $x_j = \hat{x}_j$. While, for other GPs, $y_j = 0$ and $x_j = 0$. After the allocation, some storage remains unused. However, we need not to reallocate it to the Y throwbox because their storage already matches their RoS.

5.3.3. *Discussion*

When $Y \geq |\hat{\mathbf{x}}|$, the joint scheme is optimal because it realizes the ideal allocation which is optimal. However, as only $|\hat{\mathbf{x}}|$ throwboxes are needed, $Y - |\hat{\mathbf{x}}|$ throwboxes remain unused. On the other hand, when $Y < |\hat{\mathbf{x}}|$, some storage is left unused. The ideal allocation is not realized and the joint scheme may not be optimal under some situations. For example, a GP that has a small coefficient $\lambda_j/\overline{S_j}$ but very large

Table 2
Simulation parameters.

Parameters	Default value
Average size of data	1 MB
Data generation interval	5 min
Simulation duration	72 h
Computation period, T	24 h
Total lifecycle of data	8 h
Storing lifecycle T_l of data	2 h
Threshold of visiting time, τ	10 min

RoS may not have a throwbox equipped according to the joint scheme. However, it is actually a better option to place a throwbox, because it can use up the unused storage to achieve a larger data delivery probability $\lambda_j \cdot x_j/\overline{S_j}$. Nevertheless, the joint scheme is still high in efficiency, since the Y selected GPs own the largest $\lambda_j \cdot \widehat{x}_j/\overline{S_j}$ under the ideal solution.

Throwboxes and storage are both network resources. The unused throwboxes or storage means oversubscribing of resource, which brings unnecessary cost. In consequence, it is necessary to find a balance between Y and X, so that these two kinds of resources can be both used out. According to Section 5.3.2, such a balance can be expressed as $Y=|\widehat{\mathbf{x}}|$. It is a step function allowing X to change within a particular range and thus is robust. Based on such a balance, the network designer can purchase throwboxes and storage with a proper proportion and avoid resource wastage. Moreover, the joint optimal scheme can be easily achieved under this balance.

6. Performance evaluation

OMNeT++ [25] based simulations are conducted to validate the efficiency of the proposed schemes. The network scenario is constructed based on the Dartmouth mobility trace, which is obtained from a 5-year experiment [21]. In the experiment, numerous Wi-Fi access points are deployed at the main buildings of the Dartmouth campus. Once a user connects/disconnects to/from an access point, this information is recorded in a log file. Through this method, the visiting history of each user is recorded. In our simulations, 64 users are randomly selected from the log and set to move according to their mobility traces. The buildings they frequently visit are regarded as the GPs for throwbox deployment and storage allocation. According to the selected trace, there are 9 GPs. A threshold of visiting time τ is used to exclude short passages. For data storing on throwboxes, the *First-Come-First-Serve* (*FCFS*) scheme is adopted. When the buffer of a throwbox is full, the *DropOldest* [26] strategy is applied for data refreshing. According to DropOldest, when a piece of data is sent to a throwbox whose buffer is full, the oldest data on the throwbox will be removed to provide space for the newly coming one, even if the data is still in its storing lifecycle. The major parameters used in the simulations are set as shown in Table 2. The total lifecycle of data indicates the time that data can stay in the network. After the time, the data will be destroyed.

6.1. Schemes in comparison

Four storage allocation schemes are compared. Wherein, two schemes are designed for the case with fixed throwboxes: the *Established-Deployment-and-Optimal-Allocation* (*EDOA*) scheme and the *Established-Deployment-and-Uniform-Allocation* (*EDUA*) scheme. The other two are designed for the case with deployable throwboxes: the *Optimal-Deployment-and-Optimal-Allocation* (*ODOA*) scheme and the *Random-Deployment-and-Uniform-Allo-cation* (*RDUA*) scheme. EDOA is the optimal storage allocation studied in Section 5.2, based on an established throwbox deployment scheme. While, in EDUA, storage is uniformly allocated to the fixed throwboxes. The deployment of throwboxes is established using the metric-based deployment scheme [17]. According to [17], throwboxes are simply deployed at the places with the largest value of particular metrics, such as betweenness centrality [27], degree centrality [28]. In the simulations, we adopt degree centrality as the metric. ODOA is our joint optimization scheme studied in Section 5.3. In RDUA, throwboxes are deployed randomly and storage is allocated to each throwbox uniformly.

In the simulations, a data source periodically generates data and sends them to a randomly chosen destination user. Two data delivery approaches – *Epidemic Routing* (*ER*) [15] and *Homing Spread* (*HS*) [9] – are employed for data delivery. ER is a flooding scheme in which every node can act as a data relay that helps storing and forwarding data. There is no restriction on the number of copies for each data. However, according to ER, data are not stored at throwboxes. We modify it and let each data relay store a copy of data at the throwboxes it passes. HS is a multi-copy scheme. The total number of copies of each data is restricted with a constant c. We set $c = 8$, which is a proper value with respect to the number of users and GPs [10].

6.2. Results and discussion

The weight α in *RoS* is a crucial factor that directly affects storage allocation. In this section, we first study the optimal setting of α and then study other performance metrics with this setting.

6.2.1. Optimal setting of weight α

The values of α directly decide the value of RoS. A small RoS may make a GP unable to get enough storage and some data are removed during their storing lifecycle. We call such a phenomenon "*data loss*". On the other hand, with a large RoS, a GP may obtain a storage larger than its actual demand and lead to a waste of storage. This will also cause data loss because some other GPs cannot get sufficient storage. Therefore, a good setting of α is important for reducing data loss.

As $T = 12T_l$ ($T = 24$ h and $T_l = 2$ h as set), we have $HRoS = 12SRoS$ and $RoS = (11\alpha + 1)SRoS = kSRoS$ where $k = (11\alpha + 1) \in [1, 12]$. In this case, we set $k = 1$ to 12 and study the optimal value of k. The results shown in Fig. 6 indicate that both ER and HS suffer a terrible data loss when k is set to be too small or too large. However, when k has a medium value, such as 5 or 6, both the two approaches achieve a much smaller data loss. Therefore, in the following simulations, we set $k = 5$, namely $\alpha = \frac{4}{11}$.

6.2.2. Data loss

The first performance metric we study is data loss, since it directly reflects the efficiency of storage allocation. A well

Please cite this article as: B. Fan et al., Optimal storage allocation on throwboxes in Mobile Social Networks, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.08.015

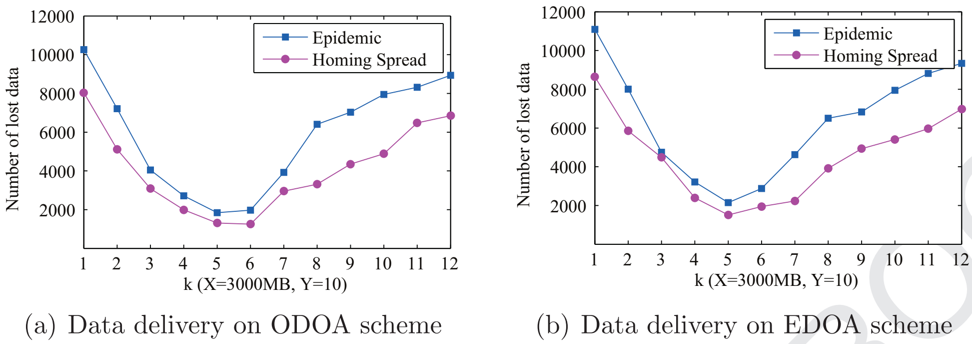


Fig. 6. Data loss under different values of k: (a) data delivery on ODOA scheme and (b) data delivery on EDOA scheme.

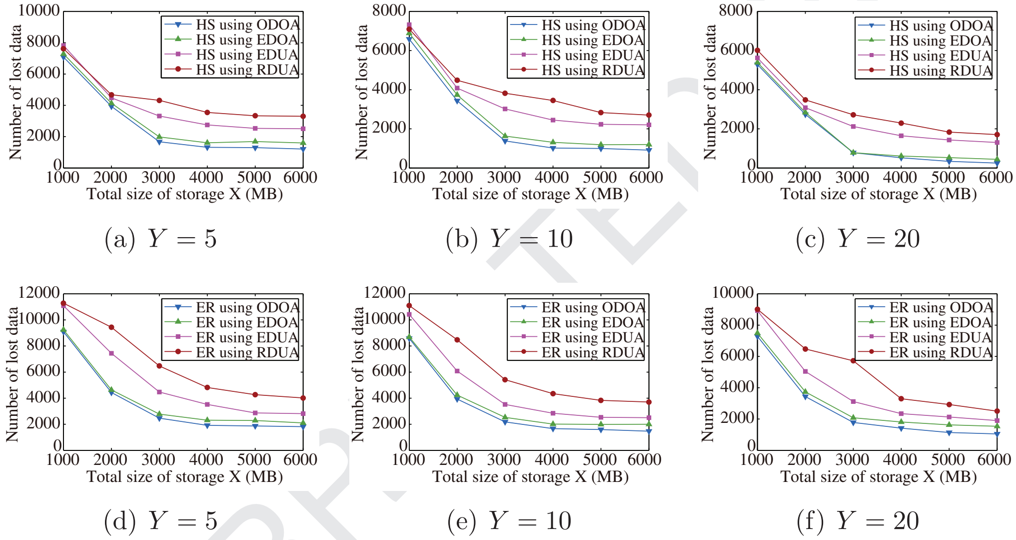


Fig. 7. Data loss of Homing Spread and Epidemic Routing: (a) $Y = 5$, (b) $Y = 10$, (c) $Y = 20$, (d) $Y = 5$, (e) $Y = 10$ and (f) $Y = 20$.

designed storage allocation scheme can properly balance the storage of each throwbox and minimize data loss. We run ER and HS using the four schemes (ODOA, EDOA, EDUA and RDUA), respectively, and compare their performances in terms of data loss.

The results are shown in Fig. 7. Not surprisingly, both ER and HS achieve the smallest data loss when using the ODOA scheme. This is because that the GPs to place throwboxes in ODOA are expressly selected for storage allocation. Therefore, storage allocation can be well performed at these GPs. When adopting EDOA, a larger data loss is suffered because EDOA adopts an individual throwbox deployment scheme [17] to deploy throwboxes. Storage allocation is not considered in the deployment. Hence, the GPs selected to place throwboxes may be not as excellent as those in ODOA for storage allocation. However, since we adopt degree centrality as the metric for throwbox deployment [17], the selected GPs are the most popular ones in the networks. Such a deployment is close to one in ODOA scheme. Consequently, although not as good, EDOA has a close performance to ODOA. When using EDUA, an even larger data loss is experienced, because storage is allocated uniformly on the fixed throwboxes. This may lead that some unpopular GPs obtain too much storage, while some popular GPs fail to get enough storage. RDUA scheme performs the worst among the four schemes, as it has low efficiency in both throwbox deployment and storage allocation.

On the other hand, ER experiences a larger data loss than HS even using the same storage allocation scheme. This is because that ER does not restrict the number of data copies. In this case, for each data, there may be more copies that need to be stored at the throwboxes. Consequently, a larger data loss is suffered when the storage is exhausted.

6.2.3. Delivery ratio

The second performance metric we study is *delivery ratio*– the ratio of data successfully reaching the destination. This is a common but important performance metric for data delivery. Hence, we adopt this metric to study the capability of these schemes in enhancing the performance of data delivery.

The results in Fig. 8 indicate that the delivery ratio of both HS and ER is improved when more throwboxes and storage are provided, since more data are able to be stored. When using the ODOA scheme, both ER and HS achieve the best delivery ratio. This is because ODOA not only balances storage allocation according to the demand of each GP, but also preferentially allocates storage to the GPs with large contact strength. In this case, even if data loss is unavoidable,

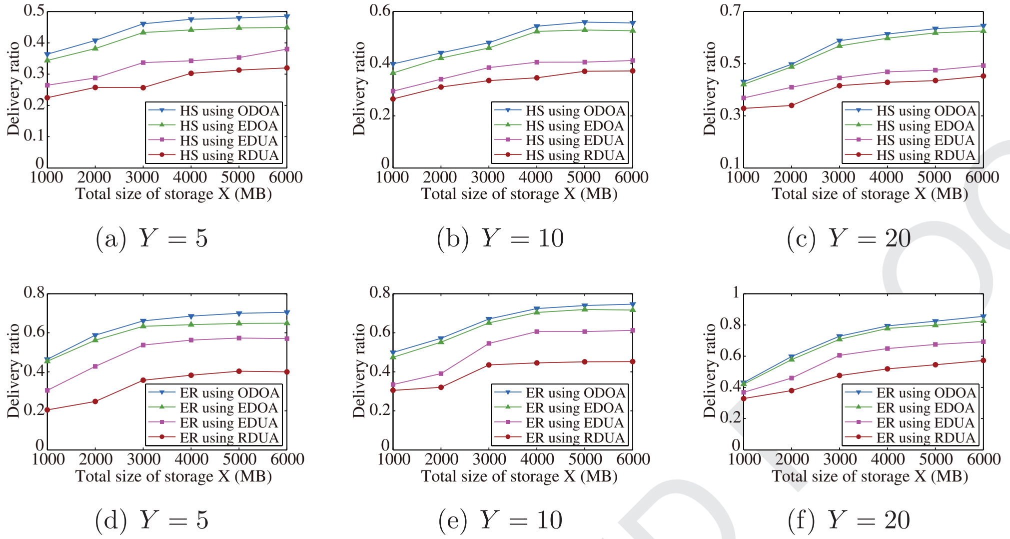


Fig. 8. Delivery ratio of Homing Spread and Epidemic Routing: (a) $Y = 5$, (b) $Y = 10$, (c) $Y = 20$, (d) $Y = 5$, (e) $Y = 10$ and (f) $Y = 20$.

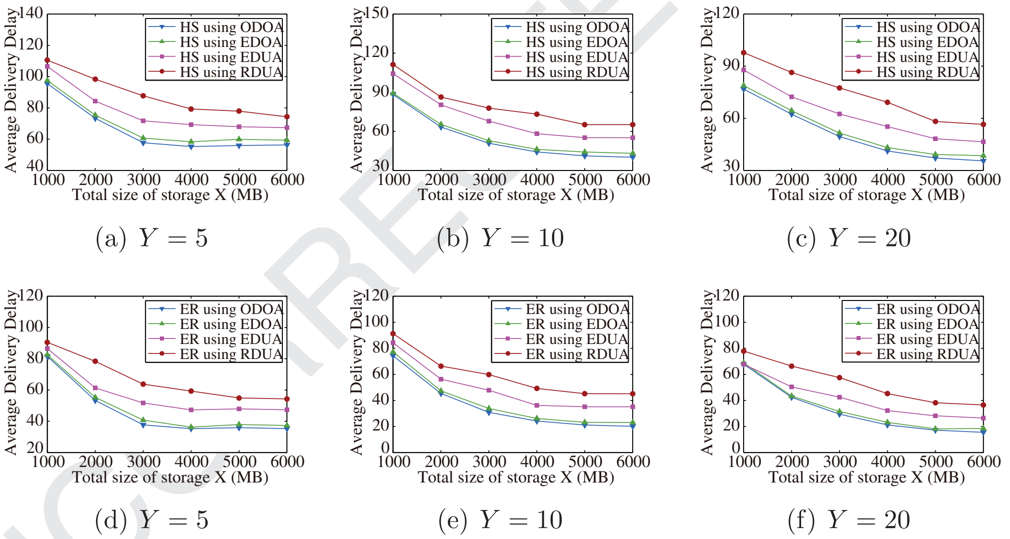


Fig. 9. Delivery delay of Homing Spread and Epidemic Routing: (a) $Y = 5$, (b) $Y = 10$, (c) $Y = 20$, (d) $Y = 5$, (e) $Y = 10$ and (f) $Y = 20$.

most data loss happens at the GPs that have the smallest contact strength. As a consequence, the loss of data delivery chance is minimized. Based on EDOA, an approximate performance in delivery ratio is achieved. This is because EDOA has a similar data loss as ODOA. Moreover, the GPs to deploy throwboxes in EDOA are also popular GPs with large contact strength. Hence, data loss also usually happens at unpopular GPs. EDUA performs badly because even throwboxes are placed at the most popular GPs, the terrible data loss caused by the uniformly storage allocation will also decrease data delivery efficiency. Finally, RDUA still performs the worst because of its random deployment and uniform allocation strategy.

The comparison between ER and HS indicates that, although suffering a worse data loss, ER still achieves a larger delivery ratio than HS. This is because ER generates more data copies than HS and can achieve more data delivery chances. In other words, the unlimited data copies bring both a larger delivery ratio and a worse data loss to ER.

6.2.4. Delivery delay

Delivery delay is another critical performance metric for data delivery, especially for networks with intermittent links among nodes, such as DTN and MSN. In throwbox-aided networks, in addition to the design of data delivery approaches, delivery delay is also decided by the efficiency of the storage allocation scheme, because a large delivery delay is usually caused when data are removed before the destination user arrives. Indeed, comparing Figs. 7 and 9, it is easy to discover that delivery delay changes nearly in line with data loss under the same scenario. Among the four schemes, not surprisingly, ODOA performs the best. EDOA, EDUA and RDUA successively have a worse performance in delivery delay, in line with their performance in data loss.

7. Conclusion

The intermittent links of nodes in Mobile Social Networks (MSNs) significantly challenge the design of data delivery approaches. Recently, several researches study the utilizing of a type of storage devices called throwboxes in data delivery. With a throwbox storing data at a particular place, the efficiency of data delivery can be enhanced, as data can be successfully forwarded as long as two nodes pass a throwbox within a particular time interval. Many proposals have studied throwboxes in the context of throwbox deployment, routing designing and energy usage. However, as a storage device, the efficient storage usage of throwboxes is seldom considered by existing work. In order to fill the research gap, this paper addresses storage allocation on throwboxes.

We subdivide the storage allocation problem into two more specific problems, namely (1) when throwboxes are fixed at particular places, how to allocate storage on the throwboxes; and (2) if throwboxes are deployable, how to conduct storage allocation in combination with throwbox deployment. Contact strength among users and GPs as well as the requirement of storage of each GP are derived with the contact history of users. Then, two optimization models are proposed to solve the two storage allocation problems. Simulation results indicate that the proposed storage allocation schemes perform well in both decreasing data loss of throwboxes and enhancing the efficiency of data delivery.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61374189), the joint training Ph.D project of China Scholarship Council (CSC) (201406070033), the Program for New Century Excellent Talents in University (NCET-10-0294), China, the Fundamental Research Funds for the Central Universities (ZYGX2013J009), China, EU FP7 Project CLIMBER (PIRSES-GA-2012-318939), UK EPSRC Project DANCER (EP/K002643/1) and EU FP7 Project MONICA (GA-2011-2952220).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2015.08.015.

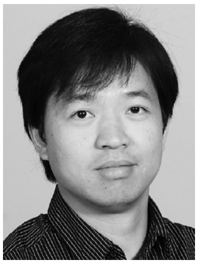
References

[1] N. Kayastha, D. Niyato, P. Wang, E. Hossain, Applications, architectures, and protocol design issues for mobile social networks: a survey, Proc. IEEE 99 (12) (2011).
[2] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of ACM Special Interest Group on Data Communication Conference, SIGCOMM, 2003.
[3] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 90–100.
[4] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, Mob. Comput.1996.
[5] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, in: Proceedings of the ACM Special Interest Group on Data Communication Workshop, SIGCOMM, 2005, pp. 252–259.
[6] P. Costa, C. Mascolo, M. Musolesi, G.P. picco, Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks, IEEE J. Sel. Areas Commun. 26 (5) (2008).
[7] E.M. Daly, M. Haahr, Social network analysis for routing in disconnected delay-tolerant manets, in: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, New York, 2007.
[8] P. Hui, J. Crowcroft, E. Yoneki, Bubble rap: social based forwarding in delay tolerant networks, in: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc, 2008.
[9] M. Xiao, J. Wu, L. Huang, Community-home-based multi-copy routing in mobile social networks, IEEE Trans. Parallel Distrib. Syst. (2014).
[10] B. Fan, S. Leng, K. Yang, Q. Liu, GPS: a method for data sharing in mobile social networks, in: Proceedings of IFIP Networking Conference, 2014.
[11] M. Xiao, J. Wu, L. Huang, Community-aware opportunistic routing in mobile social networks, IEEE Trans. Comput. (2013).
[12] M. Ibrahim, P. Nain, I. Carreras, Analysis of relay protocols for throwbox-equipped DTNs, in: Proceedings of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOPT, 2009.
[13] W. Zhao, Y. Chen, M. Ammar, M.D. Comer, B.N. Levine, E. Zegura, Capacity enhancement using throwboxes in DTNs, in: Proceedings of the 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS, 2006.
[14] M. Ibrahim, A.A. Hanbali, P. Nain, Delay and resource analysis in manets in presence of throwboxes, Perform. Eval. 64 (2007) 933–947.
[15] A. Vahdat, D. Becker, Epidemic Routing for Partially Connected Ad Hoc Networks. Technical Report CS-200006, Duke University, 2000.
[16] R. Groenevelt, P. Nain, G. Koole, The message delay in mobile ad hoc networks, Perform. Eval. 62 (2005).
[17] Z. Ying, C. Zhang, Y. Wang, Social based throwbox placement in large-scale throwbox-assisted delay tolerant networks, in: Proceedings of IEEE International Conference on Communications, ICC, 2014.
[18] N. Banerjee, M.D. Corner, B.N. LevineAn, Energy-efficient architecture for dtn throwboxes, in: Proceedings of IEEE International Conference on Computer Communications, INFOCOM, 2007.
[19] N. Banerjee, M.D. Corner, B.N. Levine, Design and field experimentation of an energy-efficient architecture for dtn throwboxes, IEEE/ACM Trans. Netow. 18 (2) (2010).
[20] B. Fan, S. Leng, C. Shao, Y. Zhang, K. Yang, Joint optimization of throwbox deployment and storage allocation in mobile social networks, in: Proceedings of IEEE International Conference on Communications, ICC, 2015.
[21] D. Kotz, T. Henderson, I. Abyzov, J. Yeo, Crawdad data set dartmouth/campus (v. 2007-02-08). Downloaded from: http://crawdad.org/dartmouth/campus/, Feb. 2007.
[22] P.V. Marsden, K.E. Campbell, Measuring tie strength, Soc. Forces 63 (2) (1984) 482–501.
[23] F. Li, J. Wu, LocalCom: a community-based epidemic forwarding scheme in disruption-tolerant networks, in: Proceedings of IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, SECON, 2009.
[24] R.K. Sundaram, A First Course in Optimization Theory, Cambridge University Press, 1996.
[25] A. Varga, The OMNeT++ discrete event simulation system, in: Proceedings of European Simulation Multiconference, ESM, Prague, 2001.
[26] D. Pan, Z. Ruan, X. Liu, Z. Song, Buffer management policies in opportunistic networks, EURASIP J. Wirel. Commun. Netw. (2013).
[27] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004).
[28] L.C. Freeman, Centrality in social networks: conceptual clarification, Soc. Netw. 1 (1979) 215–239.

Bo Fan received the B.Eng. degree in communication engineering from University of Electronic Science and Technology of China. He is currently working toward the Ph.D. degree with University of Electronic Science and Technology of China, Chengdu, China. His research interest is resource allocation and data delivery in mobile social networks.

Supeng Leng received the B.Eng. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1996 and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2005. He is an Associate Professor with the National Communication Laboratory, UESTC. He has experience as an R&D Engineer in the field of computer communications and as a Research Fellow with the Network Technology Research Center, NTU. His research focuses on ad hoc/sensor networks, wireless mesh networks, ultrawideband networks, and cognitive radio networks.

Kun Yang received his Ph.D. from the Department of Electronic & Electrical Engineering of University College London (UCL), UK. He is currently a full Professor in the School of Computer Science & Electronic Engineering, University of Essex, UK. Before joining in University of Essex at 2003, he worked at UCL on several European Union (EU) research projects for several years. His main research interests include heterogeneous wireless networks, fixed mobile convergence, pervasive service engineering, future Internet technology and network virtualization, cloud computing. He manages research projects funded by various sources such as UK EPSRC, EU FP7 and industries. He has published 50+ journal papers. He serves on the editorial boards of both IEEE and non-IEEE journals. He is a Senior Member of IEEE and a Fellow of IET.



Yan Zhang received his Ph.D. degree from Nanyang Technological University, Singapore. From August 2006, he has been working with Simula Research Laboratory, Norway. He is currently a senior Research Scientist at Simula Research Laboratory, Norway. He is an adjunct Associate Professor at the University of Oslo, Norway. He is a regional editor, associate editor, on the editorial board, or guest editor of a number of international journals. His research interests include wireless networks and smart grid communications.

Please cite this article as: B. Fan et al., Optimal storage allocation on throwboxes in Mobile Social Networks, Computer Networks (2015), http://dx.doi.org/10.1016/j.comnet.2015.08.015