J. Parallel Distrib. Comput. 86 (2015) 1–15

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

A Knapsack-based buffer management strategy for delay-tolerant networks☆

En Wang [a,*], Yongjian Yang [a], Jie Wu [b]

[a] Department of Computer Science and Technology, Jilin University, Changchun, Jilin 130012, China
[b] Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, United States

HIGHLIGHTS

- We propose a theoretical message scheduling and drop strategy in DTNs.
- We improve the theoretical strategy into a practical scheduling and drop strategy.
- We propose an efficient local parameter collection method to estimate global parameters.
- The proposed strategy utilizes the knapsack model to deal with messages in different sizes.
- We conduct extensive simulations on both synthetic and real mobility traces.

ARTICLE INFO

Article history:
Received 21 January 2015
Received in revised form
26 May 2015
Accepted 26 July 2015
Available online 3 August 2015

Keywords:
DTNs
Congestion
Knapsack problem
Scheduling
Drop

ABSTRACT

In delay-tolerant networks, the dramatic change of topology and the frequent interruption of connections make it difficult to forward the message to destination. Routing protocols in DTNs seek to improve the delivery ratio through increasing the number of message copies. However, the redundant message copies easily cause the occurrence of buffer's overflowing. In this paper, in order to maximize the utilization of network resources, especially when the bandwidth is limited and the message sizes are different, we present a theoretical framework called the Knapsack-based Message Scheduling and Drop strategy in Theory (KMSDT) based on Epidemic routing protocol. KMSDT sorts the messages in the buffer according to the per-unit utility and, if buffer overflows, decides which message to drop based on the solution to the knapsack problem. Furthermore, a practical framework called the Knapsack-based Message Scheduling and Drop strategy in Practice (KMSDP) is also developed. Rather than collecting the global statistics as done in KMSDT, KMSDP estimates all the parameters through the locally-collected statistics. Simulations based on both synthetic and real mobility traces are done in ONE. Results show that, without affecting the average delay and overhead ratio, KMSDP and KMSDT achieve better delivery ratio than other congestion control strategies.

1. Introduction

Delay-tolerant networks (DTNs) [4,9], are a kind of challenged networks in which end-to-end transmission latency may be arbitrarily long due to the occasional connections. An available connected source-to-destination path may not exist anytime. DTNs have been proposed to be used in interplanetary networks [1,33], battlefields [16], disaster response networks [26], rural areas [21], wildlife tracking [12], and pocket-switched networks [31,30]. Fall [9] puts forward this new network concept in SIGCOMM03. The nodes in DTNs exchange their messages when they encounter each other. Successful delivery occurs only when one or more infected nodes encounter the destination node.

Recently, the topology of mobile ad hoc network is regarded as connected graph, through which the end-to-end paths can be established. However, DTNs are occasionally connected and end-to-end paths are commonly unavailable due to the mobility of nodes and instability of links. A bundle layer including the store–carry–forward paradigm [5] and the custody-transfer thought is proposed to solve the above problems. It requires the node to carry a bundle and forward it to a reliable hop until the time-to-live (*TTL*) of the bundle expires. With this in mind,

☆ A conference version of the paper has appeared in Proceedings of EWSN 2015.

* Corresponding author.
E-mail addresses: wangen0310@126.com (E. Wang), yyj@jlu.edu.cn (Y. Yang), jiewu@temple.edu (J. Wu).

http://dx.doi.org/10.1016/j.jpdc.2015.07.008
0743-7315/

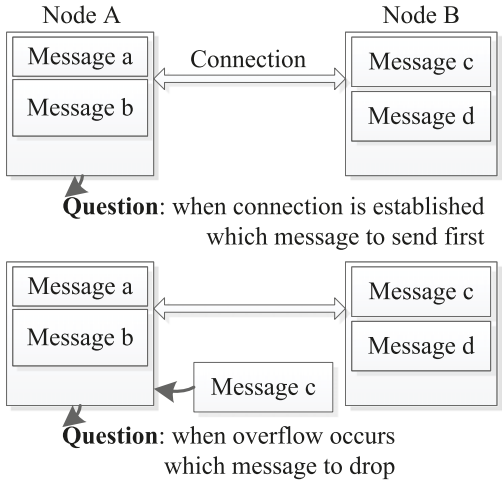


Fig. 1. An illustration of the message scheduling and drop problem.

choosing the suitable nodes to forward messages is the key point in DTNs. Since the traditional connection-based routing protocol is not available any more, many research efforts [34,24] have focused on developing effective routing protocols in DTNs. However, the proposed routing protocols seek to improve delivery ratio through increasing the number of message copies, and store the message until it finds an available link to use. The overhead in restricted bandwidth and the overflowing in limited buffer space are often neglected [29]. Especially in the realistic network environment, the congestion problem becomes more obvious.

The message scheduling and drop problem is illustrated in Fig. 1. When node *A* whose buffer stores messages *a* and *b* encounters node *B*, node *A* need to decide which message to send first. Similarly, when the buffer of node *A* overflows, it should decide which message to drop among the already stored messages (messages *a* and *b*) and the new comer (message *c*). In order to deal with these buffer-management problems, we present the Knapsack-based Message Scheduling and Drop strategy in Theory (KMSDT) and applies it on DTNs under Epidemic routing. First of all, KMSDT calculates the utility value of each message through evaluating the impact of either replicating or dropping a message on delivery ratio. Secondly, KMSDT sorts the messages according to their per-unit utilities, and whether or not to drop the message is also decided based on the solution to the knapsack problem. However, to calculate the utility, KMSDT requires some global information of the network. Therefore, it is difficult to directly implement KMSDT in practice, especially in the intermittently-connected networks. To overcome the problem, we finally propose another strategy called the Knapsack-based Message Scheduling and Drop strategy in Practice (KMSDP), which is a distributed algorithm and estimates the global network parameters through locally collected statistics. Simulations based on synthetic and real mobility traces are done in ONE, and results show that KMSDP and KMSDT achieve better performance than other congestion control strategies in terms of delivery ratio.

In this paper, we argue that GBSD [15], which adopts the non-heuristic strategy and unifies the scheduling and drop problems, is state of the art. However, despite the elegance of the approach, GBSD has the following drawbacks: (1) It neglects the failure of transmission due to the limited bandwidth. (2) When messages of different sizes coexist in the network, GBSD drops the message with the lowest utility; however, this strategy is not necessarily optimal. (3) When calculating the utility of a given message *i*, GBSD does not consider the impact of the copies of message *i*, which will be generated in remaining *TTL* on the delivery ratio. (4) The method of estimating the parameters is not applicable when message sizes differ.

The proposed KMSDT calculates the probability of successful delivery in limited-bandwidth situations according to the contact-time distribution. When buffer overflows, KMSDT maximizes the delivery ratios of messages in different sizes based on the solution to knapsack problem. Furthermore, we improve the original GBSD utility-calculation model by taking into consideration of more copies of a given message being created in the future. In addition, the network parameters are collected independently for messages of different sizes. Through the above methods, the four drawbacks of GBSD are solved.

The main contributions are summarized as follows:

(1) We propose a message scheduling and drop strategy in theory (KMSDT) based on the improvement of GBSD [15]. It calculates the probability of successful delivery for limited-bandwidth situations according to the contact-time distribution. When buffer overflows, KMSDT maximizes the global delivery ratio of messages in different sizes based on the solution to knapsack problem.
(2) We enhance the KMSDT into a knapsack-based message scheduling and drop strategy in practice (KMSDP) through collecting the local network parameters independently for messages in different sizes.
(3) We conduct extensive simulations on both synthetic and real mobility traces. The results show that KMSDP and KMSDT achieve a better delivery ratio than other congestion control strategies without affecting the average delay or overhead ratio.

The remainder of the paper is organized as follows. We introduce the related work in Section 2. The knapsack-based scheduling and drop strategy in theory (KMSDT) and in practice (KMSDP) are presented in Sections 3 and 4, respectively. In Section 5, we evaluate the performance of KMSDT and KMSDP through extensive simulations. We conclude the paper in Section 6.

2. Related work

The classical Epidemic routing protocol [27] utilizes the database replication technology to disseminate messages in DTNs. Through Epidemic routing protocol, the random pair-wise exchanges of messages ensure eventual message delivery. After sufficient exchanges, each non-isolated node eventually receives all the messages and end-to-end packet transmission is enabled. Several simple drop strategies for Epidemic routing are listed as follows: (1) drop front (DF) [20], in which the message of longest queueing time in the buffer is dropped, when the buffer overflows; (2) drop last (DL), in which the last message received in the buffer is dropped; (3) drop oldest (DO), in which the message in the buffer with the smallest remaining *TTL* is dropped; and (4) drop youngest (DY), in which the message in the buffer with the largest remaining *TTL* is dropped. DF and DO achieve better performances compared with the other two strategies. However, researchers have also proposed some more refined buffer-management strategies: Lindgren and Phanse [18] assess a series of congestion-control strategies based on heuristic ideas, and confirm that combining the buffer-management strategy and routing protocol can achieve more efficient utilization of the network resources. Wahidabanu and Fathima [28] sort messages into different queues according to their priorities. When buffer overflowing occurs, the messages with lower priorities are dropped. To reduce the impact of dropping action on the global network performance, Dohyung et al. [6] drop the messages with the largest expected number of message copies. Erramilli and Crovella [8] present a strategy to schedule the messages according to their priorities, which are calculated based on the distance from source to destination. They also notice the lack of a non-heuristic

buffer-management model in DTNs. In [10], they propose a Social Selfishness Aware Routing (SSAR) algorithm, which considers both users willingness to forward and their contact opportunity, resulting in a better forwarding strategy than purely contact-based approaches. However, there are following drawbacks of SSAR compared with our proposed schemes. (1) SSAR focuses on choosing the appropriate node for forwarding message, and takes the node's willingness and contact probability with destination into consideration, while neglecting the message's own attributes. (2) They assume each node has unlimited buffer for its own packets, but limited buffer for others, which is not reasonable in DTNs. (3) The SSAR adopts the heuristic algorithm, but our scheme uses the non-heuristic strategy, which could dynamically adapt to the changing topology in DTNs. (4) SSAR is a node-centered strategy, while our scheme is message-centered strategy, which means that SSAR could not address the following situation: a node encounters another high-utility node, while it forwards a low-utility message to the encounter. In [17], they propose methods to eliminate unnecessary forwarding redundancy and ensure efficient utilization of network resources. The existing work is also similar with our proposed scheme, however, there are following drawbacks of existing work compared with our proposed schemes. (1) It is really difficult for a node to accurately obtain the list of messages holding by the surrounding nodes in DTNs. (2) The existing work requires the current node to choose appropriate next hop according to the list of messages holding by the surrounding nodes, in order to reduce forwarding redundancy. However, it does not consider the message copies holding by the remoter nodes. (3) The utility of a message in existing work could only eliminate the local redundancy and reflect the message's influence on local delivery ratio. However, our proposed scheme calculates the utility value of each message through evaluating the impact of either replicating or dropping a message on global delivery ratio. (4) The existing work adopts the heuristic algorithm, but our scheme uses the non-heuristic strategy, which could dynamically adapt to the changing topology in DTNs.

All the above buffer-management strategies adopt the heuristic algorithm and cannot dynamically adapt to the changing topology in DTNs. Therefore, the solutions for certain performance metrics (such as delivery ratio, average delay) are suboptimal. Some research groups have tried to develop a non-heuristic buffer-management strategy by dynamically collecting the network parameters, and then they obtain the optimal solutions for delivery ratio or average delay. For example, Yong and Meng [32] propose an adaptive and idealized buffer-management strategy considering both the limited bandwidth and the different message sizes. However, this strategy assumes that network parameters can be collected through the control channel and the unfinished transmission of messages can be continued utilizing the next opportunity; with this in mind, the assumptions are impractical in DTNs. Elwhishi et al. [7] propose a scheduling scheme for Epidemic routing and two-hop forwarding. They obtain the optimal solution for delivery ratio and average delay by solving the relevant ordinary differential equations. They assume that all messages in the network are of the same size and the method used to collect the network parameters does not achieve accurate values. In addition, they do not consider the impact of bandwidth on delivery ratio. Krifa and Barakat have published three papers in this research area. In [14], to optimize delivery ratio, they estimate the utility value of a given message by calculating the impact of replicating or dropping the message on delivery ratio, and then they drop the message with the smallest utility. Based on the result of [14], the work in [13] enhances the scheduling strategy and prioritizes the messages with the highest utility. Taking into account the strategy proposed in [14], excessive information must be stored and exchanged, which easily leads to the overload of bandwidth, Krifa and Barakat [15] propose an idealized scheduling and drop strategy called the global knowledge-based scheduling and drop (GBSD) strategy, in which signal overhead is reduced by optimizing the storage structure and statistics-collection method.

3. Knapsack-based scheduling and drop strategy

To deliver a clear problem formulation and gain useful strategy insights, we first introduce the assumptions related to the work, and put forward the congestion control problem to be addressed. Next, we identify the utility-calculation model to be used for a given message by quantifying the influence of replicating or dropping the message on delivery ratio. When buffer overflows, we decide which message to drop based on its utility and the solution to knapsack problem.

3.1. Assumptions and problem description

In this paper, we make the following assumptions about the network environment. Each message has a given *TTL*, after which the message is no longer useful and should be dropped. Subsequently, it arbitrarily chooses the source and destination as well as its size within the specified scope; messages of different sizes can coexist in the network. Neither an immunization strategy nor an acknowledgment mechanism is used to confirm the receipt of packets. The bandwidth is limited and message transmission time cannot be ignored. If the transmission of a certain message is interrupted, the message must be retransmitted. The node pairs' intermeeting times and contact durations under the mobility patterns such as random walk, random waypoint and random directions tail off exponentially [32,23,25].

This paper primarily addresses the following two problems in terms of buffer-management: (1) when more than one message exists in a node's local buffer and the node does not know whether the contact will last long enough to forward all the messages. To maximize the delivery ratio, we need to decide which message to send first. (2) If a new message arrives at a node's buffer filled with messages, in order to maximize the delivery ratio we should decide which message to drop between the messages already in the local buffer and the new comer.

To address the above two problems, we propose an idealized scheduling and drop strategy KMSDT, which first expresses the delivery ratio as a function of dynamic network parameters. The per-message utility is derived from the marginal value of the delivery ratio. Several factors including message size and bandwidth limitation are considered in this process. If the bandwidth is insufficient to forward all the messages in its local buffer, the node should replicate messages in decreasing order of their per-unit utility. If buffer overflowing occurs, the node need to decide which message to drop based on its utility and the solution to knapsack problem, in order to maximize the total utility of all the local messages.

We first define several notations. P_i is the probability that message i can be successfully delivered, $K_{(t)}$ is the total number of different messages within the network, and P is the delivery ratio of the network, then $P = \sum_{i=1}^{K_{(t)}} P_i$. n_i is denoted as the copy number of message i. The utility value U_i of a certain message i is quantified by the influence of dropping or replicating the message on delivery ratio, then we face the following three situations:

$$\begin{cases} \Delta(n_i) = 1 & \text{If replicate message } i \text{ during contact.} \\ \Delta(n_i) = 0 & \text{If no action for message } i \text{ is taken.} \\ \Delta(n_i) = -1 & \text{If drop an already existing message } i. \end{cases}$$

Therefore, $\Delta P = \sum_{i=1}^{K_{(t)}} \frac{\partial P_i}{\partial n_i} \Delta(n_i) = \sum_{i=1}^{K_{(t)}} U_i \Delta(n_i)$, $U_i = \frac{\partial P_i}{\partial n_i}$.

Table 1
The main notations through the paper.

Variable	Description
N	Total number of nodes in the network minus one
$K_{(t)}$	Number of distinct messages in the network at time t
TTL_i	Initial time-to-live (TTL) for message i
R_i	Remaining time-to-live(TTL) for message i
T_i	Elapsed time for message i since its generation time ($T_i = TTL_i - R_i$)
$n_i(T_i)$	The copy number of message i in the network after the elapsed time T_i
$m_i(T_i)$	Number of nodes (excluding source) that have seen message i from its creation time to the elapsed time T_i
E_1	Average intermeeting time between nodes
λ_1	Parameters in the exponential distribution of intermeeting time ($\lambda_1 = \frac{1}{E_1}$)
E_2	Average contact duration times between nodes
λ_2	Parameters in the exponential distribution of contact duration time ($\lambda_2 = \frac{1}{E_2}$)
M_i	Size of message i
W	Bandwidth of the contacts between nodes in a pair
U_i	Utility of message i
P_{T_i}	Probability that message i has been successfully delivered at the present moment
P_{R_i}	Probability that undelivered message i will reach the destination within time R_i
ε_i	Probability that message i can be forwarded successfully during contact
P_i	Probability that message i can be successfully delivered
P	Delivery ratio

Based on the above analyses, we sort the messages in decreasing order of per-unit utility, and messages with the highest per-unit utility are replicated first during contact. When buffer overflows, we drop the messages based on the solution to knapsack problem, in order to maximize the total utility in the local buffer.

3.2. Utility-calculation model

In DTNs, messages are forwarded through random pairwise encounters. Thus, the intermeeting time and contact duration between the nodes in a pair will influence the delivery ratio. Here we define the intermeeting time and contact duration as follows:

Definition 1. Intermeeting time: the elapsed time from the end of the previous contact to the start of the next contact between nodes in a pair.

Definition 2. Contact duration: the duration time during which the nodes in a pair stay in each other's communication range.

As mentioned in the above assumptions, the recent research shows that intermeeting times and contact durations are exponentially distributed under many popular mobility patterns such as random walk, random waypoint, and random direction. Our simulations are based on two mobility scenarios: a synthetic one (the random-waypoint mobility pattern) and real-world mobility trace (epfl, which tracks 500 taxis in San Francisco over 30 days, without loss of generality, we use the data of the first 100 taxis in this paper). Thus, we first perform simulations on the distributions of the intermeeting times and contact durations and check whether they can match an exponential distribution.

As can be seen in Fig. 2, the intermeeting times and contact durations follow approximately an exponential distribution for the two scenarios:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & x \le 0. \end{cases}$$

Suppose that λ_1 and λ_2 are the parameters for the exponential distribution of intermeeting time and contact duration, respectively. E_1 and E_2 represent the mathematical expectation values, respectively; then $\lambda_1 = \frac{1}{E_1}$ and $\lambda_2 = \frac{1}{E_2}$ (Table 1).

Our goal is to express the probability P_i as a function form of n_i, and then calculate the utility of message i by quantifying the effect of replicating or dropping a copy of message i on P_i. To achieve this goal, some probability notations used throughout the paper are defined in Table 1.

The probability for message i to be delivered is given by the probability that message i has been delivered and the probability that message i has not yet been delivered, but will be delivered during the remaining time R_i. Therefore, P_i can be formulated as Eq. (1).

$$P_i = (1 - P_{T_i})P_{R_i} + P_{T_i}. \tag{1}$$

Because of the fact that all the nodes including the destination have an equal chance to see the message i, therefore, P_{T_i} can be written as follow:

$$P_{T_i} = \frac{m_i(T_i)}{N}, \tag{2}$$

P_{R_i} as shown in Eq. (1) denotes the probability that the message i, which has not yet reached its destination, can be delivered within the remaining time R_i. Calculating P_{R_i} is the core of the utility-calculation model. In [15], P_{R_i} is approximated by the probability that one copy of message i (the total number of copies at time T_i is $n_i(T_i)$) has been delivered within time R_i, without considering that (1) more copies of message i may be created in the future and (2) the failure of message transmission may be caused by limited bandwidth. To overcome these two drawbacks, we present an improved method to calculate P_{R_i}.

We first consider the change of $n_i(t)$ along with the time t. Based on the ordinary-differential-equation model used in [2], the following relationship is derived:

$$\frac{dn_i(t)}{dt} = \varepsilon_i \lambda_1 n_i(t)[N - n_i(t)], \tag{3}$$

where ε_i is the probability that message i can be forwarded successfully during contact, as illustrated in Table 1. The parameter $\lambda_1 = \frac{1}{E_1}$ is the reciprocal of the average intermeeting time. Furthermore, λ_1 is the average number of contacts between nodes per-unit time. Therefore, after solving Eq. (3) with the initial condition $n_i(0)$, we get following equation:

$$n_i(t) = \frac{N n_i(0)}{n_i(0) + [N - n_i(0)]e^{-\varepsilon_i \lambda N t}}. \tag{4}$$

Suppose that the current time is T_i, the number of nodes that hold message i in buffers after time R_i can be expressed as Eq. (5).

$$n_i(T_i + R_i) = \frac{N n_i(T_i)}{n_i(T_i) + [N - n_i(T_i)]e^{-\varepsilon_i \lambda N R_i}}. \tag{5}$$

The parameter ε_i in Eqs. (3)–(5) can be derived from the contact-time distribution. we assume that the bandwidth is W and the size of message i is M_i. In order to successfully forward message i, the contact duration should be greater than $\frac{M_i}{W}$. In addition, the contact durations follow an exponential distribution with parameter λ_2. Therefore, ε_i can be expressed as follow:

$$\varepsilon_i = e^{-\lambda_2 \frac{M_i}{W}}. \tag{6}$$

Next, we analyze the meaning of $1 - P_{R_i}$. The equation $1 - P_{R_i}$ represents the probability that message i has not yet been delivered at T_i, and will not be delivered in the remaining time R_i ($R_i = TTL - T_i$). In other words, $1 - P_{R_i}$ gives the probability that not only the $n_i(T_i)$ nodes with message i in the buffer will not contact the destination node during R_i, but also the new infected

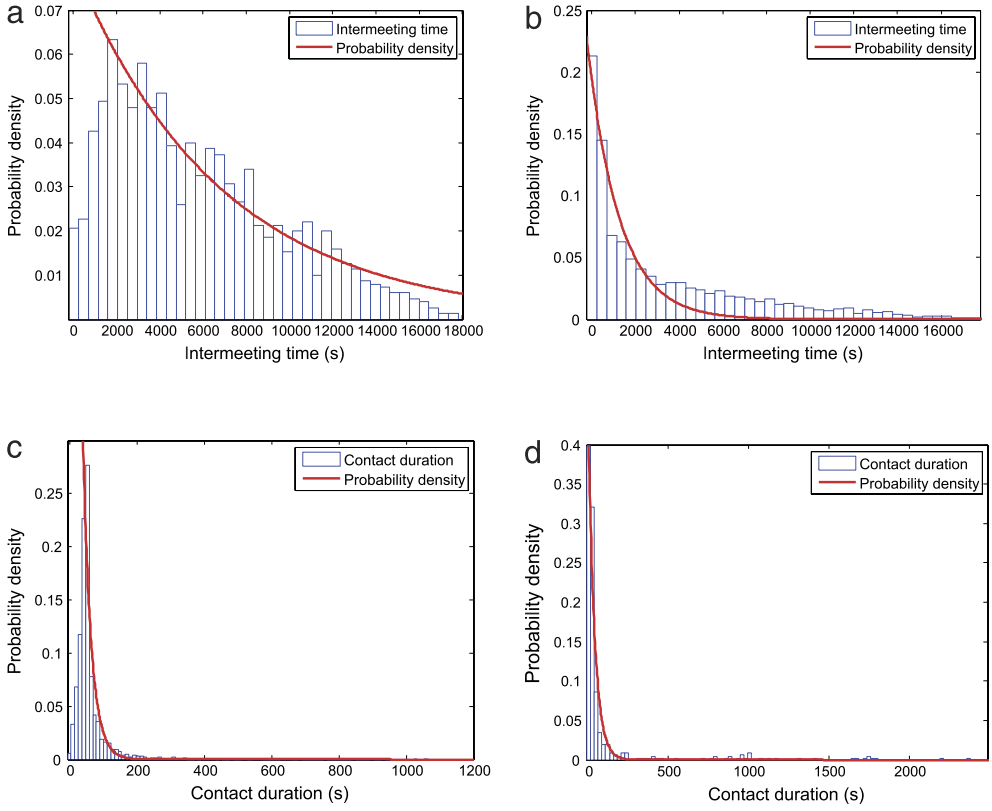


Fig. 2. Intermeeting time in (a) and (b) and contact duration in (c) and (d) distributions for the random-waypoint in (a) and (c) and real-world mobility scenarios in (b) and (d).

nodes will not finish the delivery to destination within R_i. Thus, $1 - P_{R_i}$ can be expressed as Eq. (7).

$$\begin{aligned} 1 - P_{R_i} &= e^{-\lambda n_i(T_i)R_i} \prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(R_i-t)} \\ &= \frac{e^{-\lambda n_i(T_i)R_i} \prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(R_i)}}{\prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(t)}} \\ &= \frac{e^{-\lambda n_i(T_i+R_i)R_i}}{e^{-\lambda \int_0^{R_i} n_i'(T_i+t)t\mathrm{d}t}}. \end{aligned} \tag{7}$$

The quantity $n_i(T_i + R_i)$ in Eq. (7) can be obtained through Eq. (5), and the integral in the denominator can be calculated through integration by parts:

$$\begin{aligned} &\int_0^{R_i} n_i'(T_i+t)t\mathrm{d}t = \int_0^{R_i} t\mathrm{d}n_i(T_i+t) \\ &= \left| {}_0^{R_i} tn_i(T_i+t) - \int_0^{R_i} n_i(T_i+t)\mathrm{d}t \right. \\ &= R_i n_i(T_i+R_i) \\ &\quad - \left(NR_i + \frac{\ln[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i\lambda NR_i} + Ne^{-\varepsilon_i\lambda NR_i}]}{\varepsilon_i\lambda} \right) \\ &\quad + \frac{\ln(N)}{\varepsilon_i}. \end{aligned} \tag{8}$$

By combining Eqs. (5), (7), and (8), the probability P_{R_i} that the undelivered message i at T_i can reach the destination in the remaining time R_i is shown in Eq. (9).

$$P_{R_i} = 1 - \frac{N^{\frac{1}{\varepsilon_i}}}{e^{\lambda NR_i}[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i\lambda NR_i} + Ne^{-\varepsilon_i\lambda NR_i}]^{\frac{1}{\varepsilon_i}}}. \tag{9}$$

By substituting Eqs. (2)–(9) into Eq. (1), we get the final expression for P_i:

$$\begin{aligned} P_i &= \frac{m_i(T_i) - N}{N} N^{\frac{1}{\varepsilon_i}} \\ &\quad \times \frac{1}{e^{\lambda NR_i}[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i\lambda NR_i} + Ne^{-\varepsilon_i\lambda NR_i}]^{\frac{1}{\varepsilon_i}}} + 1. \end{aligned} \tag{10}$$

Note that the delivery ratio P equals to the sum of P_i (as shown in Eq. (11)). With the help of Eq. (11), we can derive the effect of replicating or dropping a given message i on delivery ratio as Eq. (12):

$$\begin{aligned} P &= \sum_{i=1}^{K(t)} P_i = \sum_{i=1}^{K(t)} \left[\frac{m_i(T_i) - N}{N} N^{\frac{1}{\varepsilon}} \right. \\ &\quad \left. \times \frac{1}{e^{\lambda} NR_i[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i\lambda NR_i} + Ne^{-\varepsilon_i\lambda NR_i}]^{\frac{1}{\varepsilon_i}}} + 1 \right] \end{aligned} \tag{11}$$

$$\begin{aligned} \Delta(P) &= \sum_{i=1}^{K(t)} \frac{\partial P_i}{\partial n_i(T_i)} \Delta n_i(T_i) \\ &= \sum_{i=1}^{K(t)} \left\{ [N - m_i(T_i)] N^{\frac{1-\varepsilon_i}{\varepsilon_i}} e^{-\lambda NR_i} \frac{1}{\varepsilon_i} (1 - e^{-\varepsilon_i\lambda NR_i}) \right. \\ &\quad \left. \times [n_i(T_i) - n_i(T_i)e^{-\varepsilon_i\lambda NR_i} + Ne^{-\varepsilon_i\lambda NR_i}]^{\frac{-\varepsilon_i-1}{\varepsilon_i}} \Delta n_i(T_i) \right\}. \end{aligned} \tag{12}$$

The scheduling and drop strategy described in this paper aims to maximize the delivery ratio of the whole network. Whenever a given message i is replicated during contact, the copy number of message i increases by one $[\Delta n_i(T_i) = +1]$; if no operation is performed on message i, the copy number of message i remains unchanged $[\Delta n_i(T_i) = 0]$; when a copy of message i is dropped from the buffer, the copy number of message i decreases by one

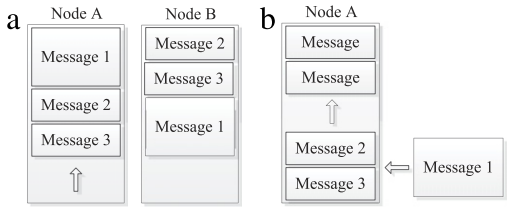


Fig. 3. Different schedule (a) and drop (b) strategies in a buffer.

$[\Delta n_i(T_i) = -1]$. Therefore, the utility of message i is precisely the derivative of the delivery ratio P. We obtain the following equation for calculating message utility:

$$U_i = [N - m_i(T_i)]N^{\frac{1-\varepsilon_i}{\varepsilon_i}} e^{-\lambda NR_i} \frac{1}{\varepsilon_i}(1 - e^{-\varepsilon_i \lambda NR_i}) \times [n_i(T_i) - n_i(T_i)e^{-\varepsilon_i \lambda NR_i} + Ne^{-\varepsilon_i \lambda NR_i}]^{\frac{-\varepsilon_i - 1}{\varepsilon_i}}. \quad (13)$$

3.3. Idealized knapsack-based scheduling and drop strategy

After calculating the message utility, the scheduling and drop strategy can be executed. A higher per-message utility indicates that replicating the message would lead to a more significant increase in the delivery ratio P. Therefore, when two nodes are within each other's communication range, messages should be replicated in decreasing order of the utility to maximize P. In addition, a higher per-message utility also means that dropping the message would lead to a more significant decrease in the delivery ratio P. When a buffer overflows, the message with the lowest per-message utility should be dropped. Previous studies [14,13,15] have proven that the above scheduling and drop strategy leads to a good performance in terms of delivery ratio. However, when message sizes differ, the strategy is no longer applicable.

In Fig. 3, the vertical rectangular box represents the local buffer of a node, and the smaller rounded rectangles represent the messages stored in the local buffer. The utilities per message satisfy: $U_1 > U_2 > U_3$, and $U_2 + U_3 > U_1$. The message sizes satisfy: $M_1 = 2M_2$ and $M_2 = M_3$. Fig. 3-(a) shows two different scheduling methods. We assume that only messages with no greater than M_1 can be successfully forwarded due to the limited contact duration and bandwidth. By analyzing the first scheduling method, only message 1 would be replicated, which leads to a gain of U_1 in the delivery ratio. However, through the second scheduling method, messages 2 and 3 would be replicated, which leads to a gain of $U_2 + U_3$ in the delivery ratio. Thus, the second scheduling strategy obtains a better performance in terms of delivery ratio.

According to the above analyses, the scheduling strategy that simply considers the per-message utility cannot be applied to the networks where message sizes differ. Additionally, the bigger message size indicates that, for a given bandwidth, more buffer space is occupied and more transmission time is required. Therefore, we schedule the messages according to the utility per-unit $\frac{U_i}{M_i}$ and replicate messages in decreasing order of $\frac{U_i}{M_i}$.

In Fig. 3-(b), the message utilities also satisfy $U_1 > U_2 > U_3$, and $U_2 + U_3 > U_1$. The message sizes satisfy $M_1 = 2M_2$ and $M_2 = M_3$. In addition, the local buffer is already full. Messages 2 and 3 are the smallest-sized messages. When message 1 arrives and the buffer overflows, which message to drop, among all buffered messages and the new-comer, should be decided. If we simply drop the message with the smallest utility, message 3 would be dropped first, followed by message 2. In this case, the delivery ratio would decrease by $U_2 + U_3$. If we adopt another strategy and only drop message 1, the delivery ratio would decrease by U_1. Thus, when message sizes differ, dropping the message of the smallest utility is obviously not the optimal strategy.

Algorithm 1 DPMKP

```
for j = 0; j ≤ totalWeight; j++ do
  for i = 0; i ≤ n; i++ do
    if (i = 0||j = 0) then
      bestValues[i][j] = 0;
    else
      if j < sizes[i − 1] then
        bestValues[i][j] = bestValues[i − 1][j];
      else
        iweight = sizes[i − 1];
        ivalue = values[i − 1];
        bestValues[i][j] = MAX(bestValues[i − 1][j]);
        ivalue = ivalue + bestValues[i − 1][j − iWeight];
if bestSolution = null then
  bestSolution = NEWint[n];
tempWeight = totalWeight;
for i = n; i ≥ 1; i−− do
  if bestValues[i][tempWeight] > bestValues[i − 1][tempWeight] then
    bestSolution[i − 1] = 1;
    tempWeight = sizes[i − 1];
  if tempWeight = 0 then
    break;
bestValue = bestValues[n][totalWeight];
```

It is worth noticing that if the buffer size is fixed, the purpose of the drop strategy is to maximize the total utility of all messages in the local buffer. Therefore, the message drop problem changes to solve the following typical 0–1 knapsack problem:

$$\text{Restriction}: \sum_{k=1}^{n} M_k x_k \le M, \quad x_k = \{0, 1\},\ k = 1, 2, 3 \cdots n,$$

$$\text{Objective}: Max \sum_{k=1}^{n} U_k x_k$$

where U_k is the utility of the k_{th} message. M is the buffer size, and M_k is the size of the k_{th} message. The number of all the buffered messages and the newly arrived message is denoted as n. x indicates whether the k_{th} message is buffered.

To solve the 0–1 knapsack problem as described above, we adopt the dynamic programming method (as shown in Algorithm 1) to decide which messages should be buffered and which to drop.

To summarize, the message utility can be calculated by Eq. (13). Because of the various message sizes, the messages are scheduled according to the per-unit utility $\frac{U_i}{M_i}$ and are replicated in decreasing order of $\frac{U_i}{M_i}$ during contact. When buffer overflowing occurs, drop decisions are made based on the solution to the 0–1 knapsack problem, in order to maximize the total utility of all the messages in the local buffer. Through this method, we propose the idealized scheduling and drop strategy KMSDT. However, in order to calculate the utility of message i through Eq. (13), $n_i(T_i)$ and $m_i(T_i)$ at time T_i must be known. However, KMSDT assumes that every message can perceive its own value of $n(T)$ and $m(T)$. The assumption in terms of global parameters indicates that KMSDT is unavailable in real-DTN scenarios. Therefore, we propose the KMSDP, which estimates the global parameters $n(T)$ and $m(T)$ by utilizing distributed collected-history information.

4. Knapsack-based scheduling and drop strategy in practice

According to the above descriptions, it is clear that the idealized scheduling and drop strategy KMSDT requires global information of the network. The research [32,2] suggests that network parameters can be obtained through the control channel.

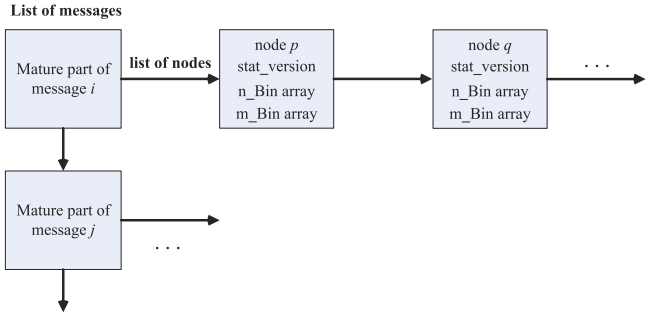


Fig. 4. Network history data structure.

However, this is not applicable in a realistic network environment. Therefore, we propose a practical knapsack-based scheduling and drop strategy KMSDP. For the reason that it is difficult to determine $n_i(T_i)$ and $m_i(T_i)$ at time T_i, each node must estimate the current $n_i(T_i)$ and $m_i(T_i)$ for message i by collecting the history statistics of the once-stored messages. Each node maintains a list of messages of which it tracks the history in the network. For each message, it maintains a list of nodes that have already seen the message, and it also store the arrays recording whether or not the message is ever buffered. This history information is used to approximate $n(T)$ and $m(T)$, which are used to calculate the utility of each message.

4.1. Collecting and maintaining history information

Fig. 4 shows the history data structure that KMSDP uses. Each node selects a part of the once-stored messages whose history data must be collected and puts them into its list of messages. Each item in the message list contains the message identification (ID) and the mature part of the message. The mature part of the message is a time field, which means that $n(T)$ and $m(T)$ before this time field are mature statistics (the meaning of mature statistics is explained in Section 4.2). Each node in the node lists maintains its node ID, stat_version, *n_Bin*[], and *m_Bin*[]. The stat_version entry represents the time unit (i.e., bin) when the last update occurs. The *n_Bin*[] entry indicates whether or not the node stores the message during the certain bin. Finally, the *m_Bin*[] entry indicates whether or not the node has ever stored a copy of the message before the bin. Both *n_Bin*[] and *m_Bin*[] are boolean arrays. For example, $n\_Bin[k] = 1$ means that the node buffers a copy of the message during bin k, and $m\_Bin[k] = 1$ means that the node ever stored a copy of the message before bin k (whether it still stores the copy is uncertain). The bin size (i.e., length of the time unit) for *n_Bin*[] and *m_Bin*[] is called *Bin_Unit*. Because the messages have a fixed *TTL*, a larger *Bin_Unit* leads to a smaller *n_Bin*[]. So the nodes need to maintain and exchange fewer data. However, with a larger *Bin_Unit*, some contact information would be omitted, which leads to an inaccurate utility calculation. For a smaller *Bin_Unit*, the node needs to maintain more data, so the utility calculation is more accurate. However, because of the limited bandwidth, exchanging a large amount of data during contact could easily makes the strategy inoperable. Consider that the values stored in *n_Bin*[] and *m_Bin*[] can only be changed when the nodes encounter with each other, the size of *Bin_Unit* depends on the average intermeeting time E_1. In order to achieve a good tradeoff between bandwidth overload and statistical accuracy, we argue that, based on [11], *Bin_Unit* should be determined as $Bin\_Unit = \frac{E_1}{2}$.

For instance, the *TTL* of the message is 18 000 s and the average intermeeting time is $E_1 = 3000$ s (obtained by the intermeeting time statistics), then $Bin\_Unit = 1500$ s and the length of *n_Bin*[] and *m_Bin*[] is 12, and we may get the following bin arrays: *n_Bin*[] = (0 0 0 0 1 1 1 0 0 0 0 0) and *m_Bin*[] = (0 0 0 0 1 1 1 1 1 1 1 1). According to the data in the bin arrays, we can learn that, during the first-four bins, the node has not stored or seen the message. And the message reaches the node during bin 5. Afterwards, it is dropped during bin 8.

According to the above considerations, we present the following two questions: (1) there are different sizes of messages, which message's statistics (taking into account the message sizes) should be used to estimate the parameters of the present message? And (2) whether the nodes need to collect the history statistics of all messages ever stored or just a part of the statistics?

To address the first question, experiments based on the random-waypoint mobility are done and the message sizes are randomly chosen from following scope (0.5, 1, 1.5, 2 MB) to obtain the change in $n_i(T_i)$, $m_i(T_i)$ and $m_i(T_i) - n_i(T_i)$ during the simulation time for a given message i. For messages of the same size, we arbitrarily select two of them and get the results as shown in Figs. 5–8.

In Figs. 5–8, the green, blue and red curves show the changes in $n(T)$, $m(T)$, and $m(T) - n(T)$, respectively. As can be seen in the results, messages of the same size have similar curves, including the changes in amplitude. In addition, the peak values are almost the identical. However, the curves for messages of different sizes are quite different. To estimate the utility of message i as accurately as possible, we use the history statistics of messages whose sizes are the same with message i to estimate $n_i(T_i)$ and $m_i(T_i)$.

To address the second question, we first point out that collecting the history information from all messages would increase the storage and bandwidth overhead. In addition, because the local buffers of the nodes are limited in size and the messages are generated arbitrarily, the nodes may monitor too many messages of the same size and miss monitoring other types of messages. Therefore, we present another data structure. We assume that there are n types of messages in the network (in our experiment $n = 4$, and message sizes are 0.5, 1, 1.5, and 2 MB) and they equally divide the local buffer (the part to store the information) into n parts so that each part can be used to monitor a fixed number of messages. When a new message arrives, we first check whether or not the corresponding buffer section used to monitor this kind of message is full. If the buffer section still has space, we add the new message to the message list and begin to collect the message's information. When the *TTL* for the message being monitored elapses, it is deleted from the message list. Through this method, different messages have an equitable chance to be monitored.

Furthermore, for each type of messages, the maximum number of messages to be simultaneously monitored should be determined. In order to not miss messages (i.e., each generated message should have an opportunity to be collected), newly generated messages of the same type should all have the opportunity to be monitored from the time that the first message of this type arrives at the buffer to the time that *TTL* elapses. Given a message generation interval: t_m and number different message sizes: n_m, the average time that messages of the same size are generated is $t_m n_m$. Therefore, $\frac{TTL}{t_m n_m}$ messages should be monitored for each type of messages.

Each node in the data structure is supposed to maintain up-to-date statistics; the detailed update operation includes the following two parts: (1) At the beginning of each *Bin_Unit*, update *n_Bin*[] and *m_Bin*[] in the lists of nodes. If a certain field within *n_Bin*[] changes, update stat_version to the current time. Otherwise, keep stat_version unchanged. (2) When the nodes encounter each other, they check whether or not they have monitored the identical messages (e.g., collect the history statistics of the same message). If so, replace the old version with the new version.

4.2. Estimation of $n(T)$ and $m(T)$

For every message being monitored, each node uses two one-dimensional arrays *n_sum* and *m_sum*, whose sizes equal to the

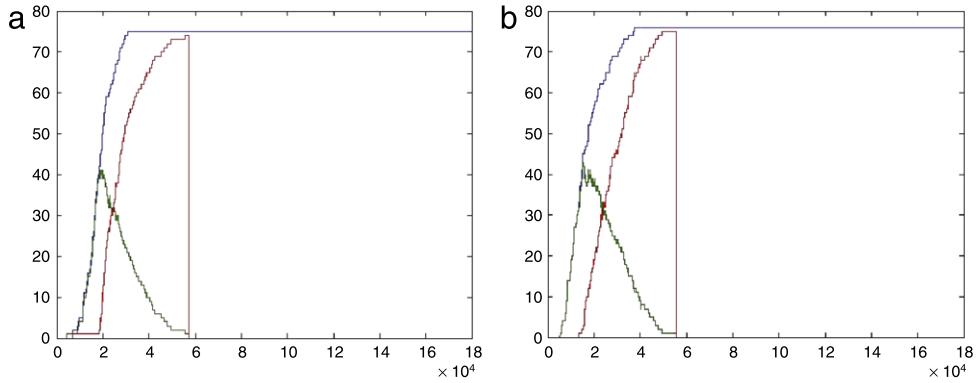


Fig. 5. Plots of $n(T)$, $m(T)$, and $m(T) - n(T)$ for 0.5 MB messages. The green curve is the change in $n(T)$, the blue curve is the change in $m(T)$, and the red curve is the change in the difference $m(T) - n(T)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

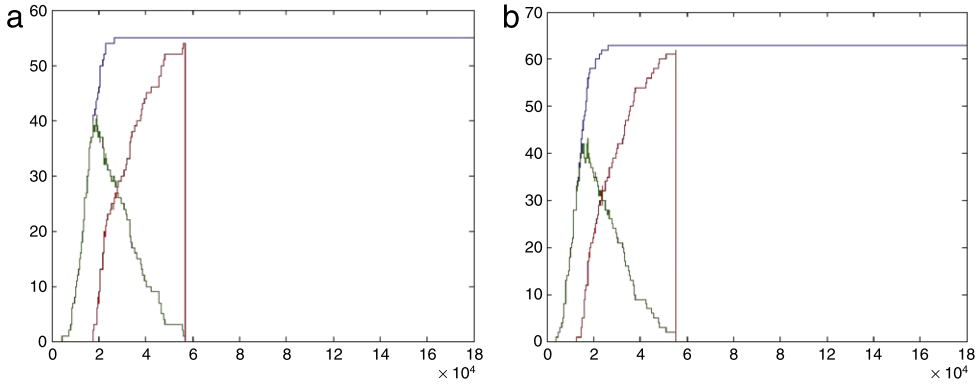


Fig. 6. Plots of $n(T)$, $m(T)$, and $m(T) - n(T)$ for 1 MB messages. The color scheme is the same as for Fig. 5. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

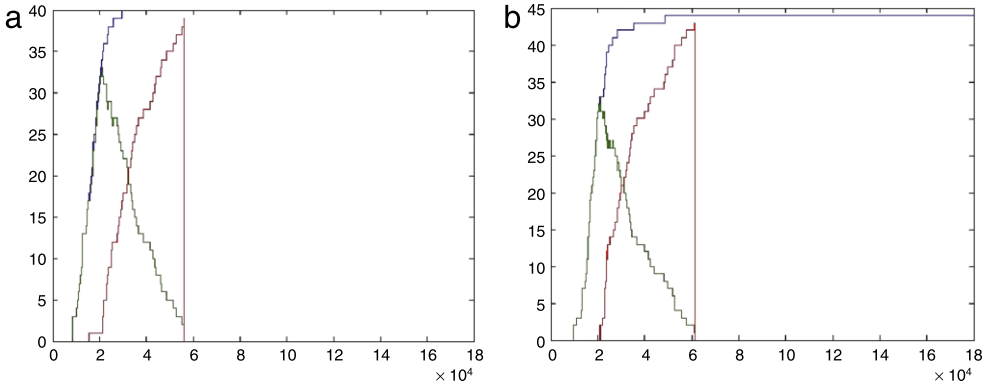


Fig. 7. Plots of $n(T)$, $m(T)$, and $m(T) - n(T)$ for 1.5 MB messages. The color scheme is the same as for Fig. 5. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sizes of $n(T)$ and $m(T)$ in the local buffer (e.g., *Bin_Number*) to record the weighted average history statistics of $n(T)$ and $m(T)$, respectively. The goal is to use the history statistics to estimate the current parameters of the newly-arrived message. However, during the collection of network parameters, some information obtained is immature because of the transmission delay in DTNs. We assume that some nodes have updated the items related to the current node in their data structure; the updated record might take a long time to reach the current node, so some parts of the information in the local data structure is incomplete and immature. Note that the average time needed for the information to reach the current node is the average intermeeting time (i.e., E_1). In this paper, we assume that the current time is T. Therefore, the history information prior to time $T - E_1$ is mature and can be used to update the mature data (*n_sum* and *m_sum*).

Whenever the information of a certain message i becomes mature (as shown in Fig. 9), we first calculate $n_i(T_i)$ and $m_i(T_i)$ for this message. Next, we calculate the mean of the newly derived $n_i(T_i)$ and $n\_sum(T_i)$ that is already stored in the array *n_sum*. We then update $n\_sum(T_i)$ with the mean value. The identical process is used to update $m\_sum(T_i)$ in the array *m_sum*. Thus, the history information stored in the one-dimension array can be more similar to the current network's information, therefore, the estimations of $n(T)$ and $m(T)$ for each message at time T can be rather accurate.

In summary, in order to collect exact values of $n_i(T_i)$ and $m_i(T_i)$ at time T_i for a given message i, each node uses the history statistics of the once-stored messages to estimate $n_i(T_i)$ and $m_i(T_i)$, and then uses the estimated value to calculate the per-message utility. Based on the above process, the practical knapsack-based scheduling and drop strategy KMSDP is achieved. Simulation results show that

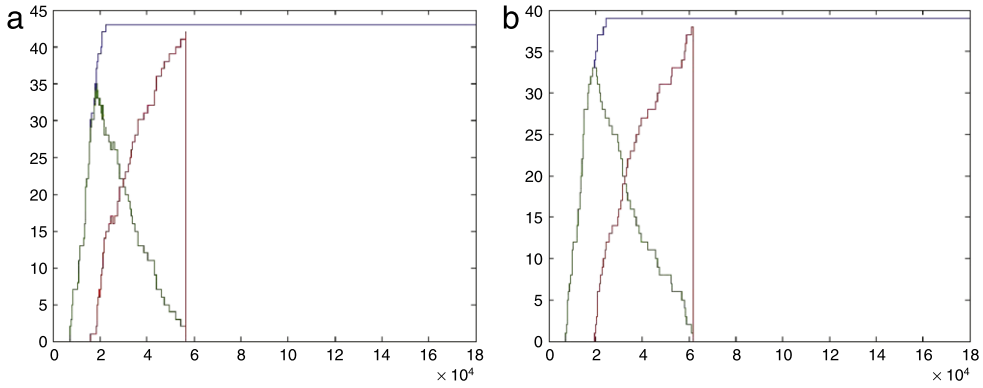


Fig. 8. Plots of $n(T)$, $m(T)$, and $m(T) - n(T)$ for 2 MB messages. The color scheme is the same as for Fig. 5. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

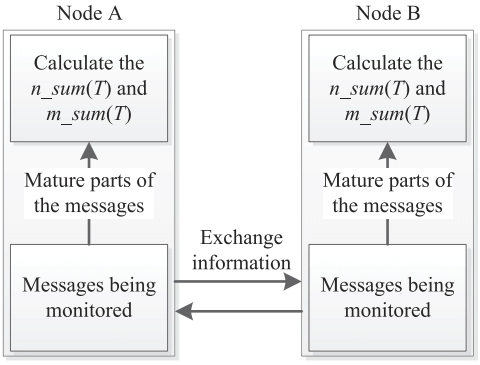


Fig. 9. Process to calculate $n\_sum(T)$ and $m\_sum(T)$.

Table 2
Simulation parameters under random-waypoint mobility pattern.

Parameter	Value
Simulation time	18 000 s
Number of nodes	100
Moving speed (random-waypoint)	2 m/s
Transmission speed	250 kbps
Transmission range	100 m
Buffer size	10, 15, 20, 25, 30 MB
Interval of message generation	[5, 15], [15, 25], [25, 35], [35, 45]
TTL	300
Message size	0.5, 1, 1.5, 2 MB

KMSDP successfully achieves similar delivery ratio compared with the idealized strategy KMSDT.

5. Performance evaluation

5.1. Experimental setup

To evaluate the performance of KMSDT and KMSDP, we use the opportunistic network simulator ONE and conduct experiments under both the synthetic random-waypoint scenario and the three real-world DTN traces: epfl [22], roma taxi [3] and pmtr [19]. In the former scenario, each node repeats its own behavior: select a destination arbitrarily and walk along the shortest path to reach the destination. In the latter real traces, epfl contains the GPS information of 500 taxis is collected for 30 days, without loss of generality, the first 100 taxis are as the input of the simulation, roma taxi contains mobility traces of taxi cabs in Rome, Italy. It contains GPS coordinates of approximately 320 taxis collected over 30 days, and pmtr contains mobility traces from 44 mobile devices at University of Milano.

To study the performance of KMSDT and KMSDP for messages in different sizes, we first determine how messages are generated. In this paper, messages are generated with the sizes arbitrarily chosen from 0.5, 1, 1.5, and 2 MB. The destination and source nodes are then chosen arbitrarily from the whole network. Next, we allow a warm-up period to collect and calculate the network parameters (without loss of generality, we set $TTL/2$ to make sure that the initial values of $n_i(T_i)$ and $m_i(T_i)$ may make KMSDP feasible). After the warm-up period, we use the Epidemic routing protocol to forward messages. Seven buffer-management strategies (KMSDT, KMSDP, GBSD [15], DF, DL, DO, and DY) are implemented in order to compare their performances. The simulation parameters are given in Table 2.

While a range of data is gathered from the experiments, we take the following three main performance metrics into consideration.

(1) *Delivery ratio* $= \frac{\textit{The number of messages successfully delivered to the destination}}{\textit{The total number of messages generated in the network}}$.

(2) *Average delay* $= \frac{\textit{The total elapsed time of the successfully delivered messages}}{\textit{The total number of successfully delivered messages}}$.

(3) *Overhead ratio* $= \frac{\textit{The successfully forwarded message number}}{\textit{The successfully delivered message number}} - 1$.

Overhead ratio (load ratio) reflects the efficiency of utilizing contacts. A high overhead ratio means that many copies of messages forwarded through contacts cannot reach the destination node during their *TTLs*, so the contacts are utilized inefficiently. Conversely, a low overhead ratio indicates that the contacts are used efficiently.

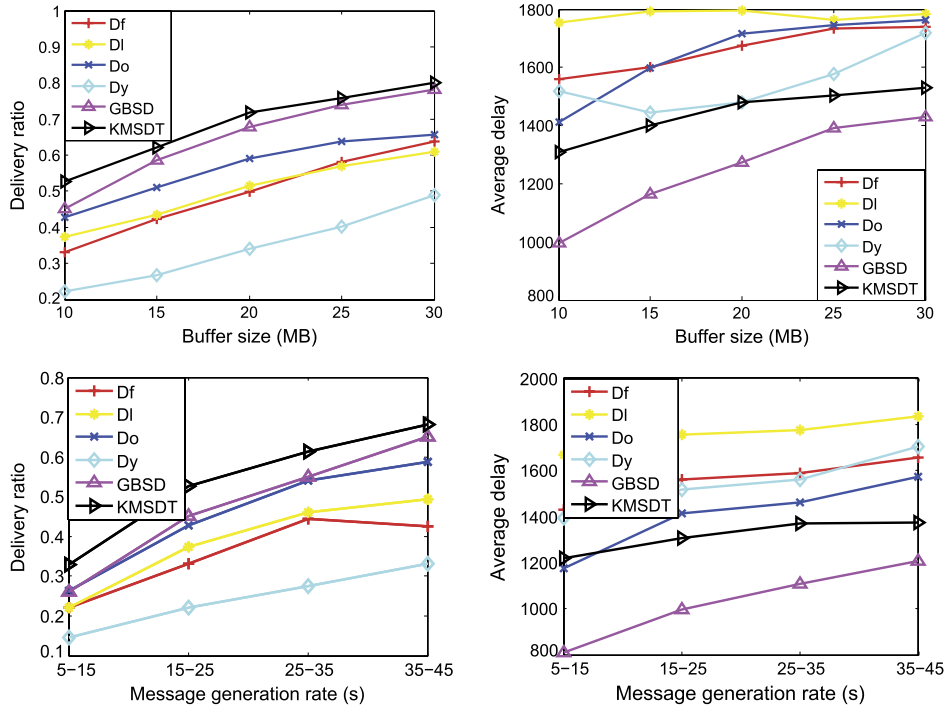
5.2. Performance analysis with the same message size

First of all, to verify the accuracy of the message utility calculated by Eq. (9), we implement the six buffer-management strategies (KMSDT, GBSD, DF, DL, DO, and DY) with the same message size of 1 MB under the synthetic random-waypoint mobility scenario. Results (as shown in Fig. 10) show that KMSDT obtains highest delivery ratio, acceptable average delay, and lowest overhead ratio in terms of different buffer size, and message generation rate, compared with other buffer-management strategies.

5.3. Performance analysis with different message sizes

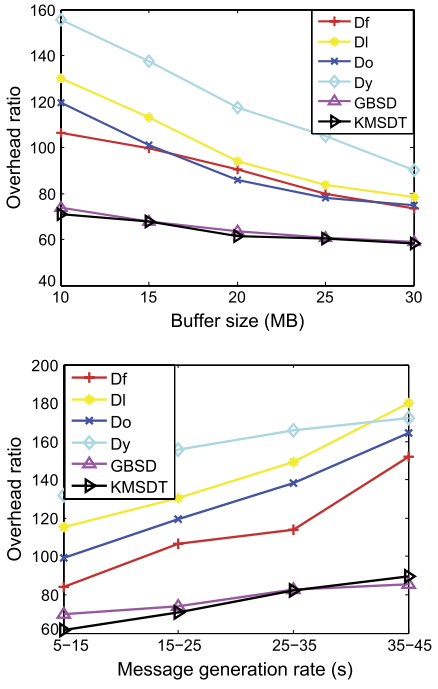
5.3.1. Performance evaluation under random-waypoint scenario

We first discuss the experiments under the synthetic random-waypoint mobility pattern. One hundred nodes are placed in a 4500×3400 m^2 area. The buffer size is 10 MB and the generation rate is one message per 15–25 s. We vary buffer size and message generation rate to evaluate the performance of KMSDT, KMSDP, and the other buffer-management strategies.



(a) Delivery ratio.

(b) Average delay.



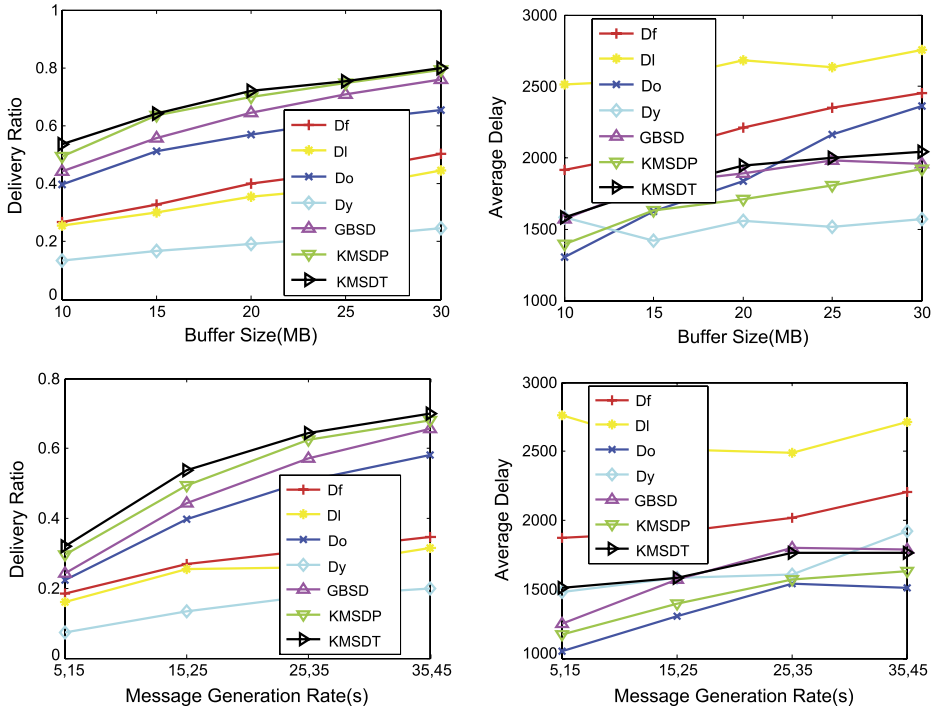
(c) Overhead ratio.

Fig. 10. Delivery ratio, average delay, and overhead ratio as a function of buffer size and message generation rate under the random-waypoint scenario (same message size).

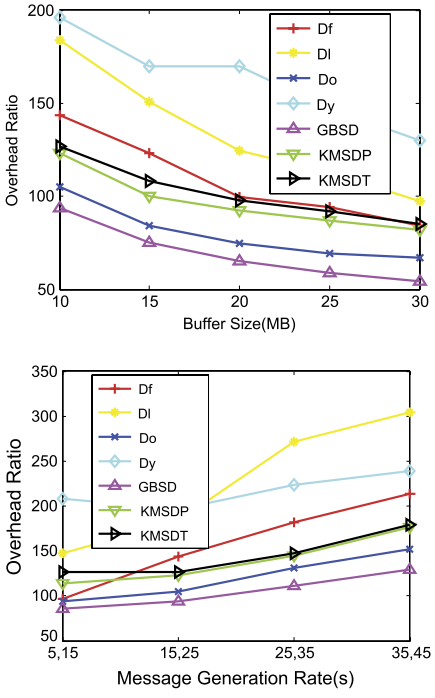
It is worth noticing that, according to the above calculation method, the calculation results of average intermeeting time and contact duration (as shown in Fig. 2) under the random-waypoint scenario are 5709.9 and 61.7 s, respectively. Therefore, we set the number of nodes to 100, and the generation rate to one message per 15–25 s. We vary the buffer sizes. The delivery ratio, average delay, and overhead ratio are plotted in Fig. 11, respectively.

Fig. 11-(a) shows that the delivery ratio increases along with the buffer size. It is mainly due to that the increasing of buffer size indicates the decreasing of the congestion level. It is worth noticing that when the buffer size increases to 30 MB, the delivery ratios of KMSDT and KMSDP reach 80%, which is much higher than the delivery ratios of the other five buffer-management strategies. Among DF, DL, DO, and DY, the DO and DF have better delivery ratios, which is natural and reasonable.

Fig. 11-(b) provides some important data regarding average delay. As can be seen, DY achieves the best performance in terms of average delay, and we attribute it to the fact that DY always drops the youngest messages (message with the smallest T_i), so the older messages are hardly delivered. The successfully-delivered



(a) Delivery ratio. (b) Average delay.



(c) Overhead ratio.

Fig. 11. Delivery ratio, average delay, and overhead ratio as a function of buffer size and message generation rate under the random-waypoint scenario.

messages under DY are always those that can be easily delivered within a short time, which leads to the lowest average delay. The average delay for KMSDP is slightly higher than that of DY, but lower than the other strategies. In addition, the average delay for KMSDT is close to that for GBSD. Therefore, KMSDT and KMSDP perform commendably in terms of average delay.

Fig. 11-(c) describes the trend of overhead ratio for different buffer-management strategies. GBSD outperforms all the other buffer-management strategies. The overhead ratios for KMSDT and KMSDP almost have no difference with that of GBSD, which is acceptable due to the fact that KMSDT and KMSDP have significantly better delivery ratios.

Next, we study how the generation rate affects the buffer-management strategies. We set the number of nodes to 100 and the buffer size to 10 MB. We vary the message generation rate and plot the delivery ratio, average delay, and overhead ratio in Fig. 11, respectively.

It is worth noticing that the notation '5,15' for the message generation rate in Fig. 11-(a) means that a new message is generated per 5–15 s. Thus, the message generation rate

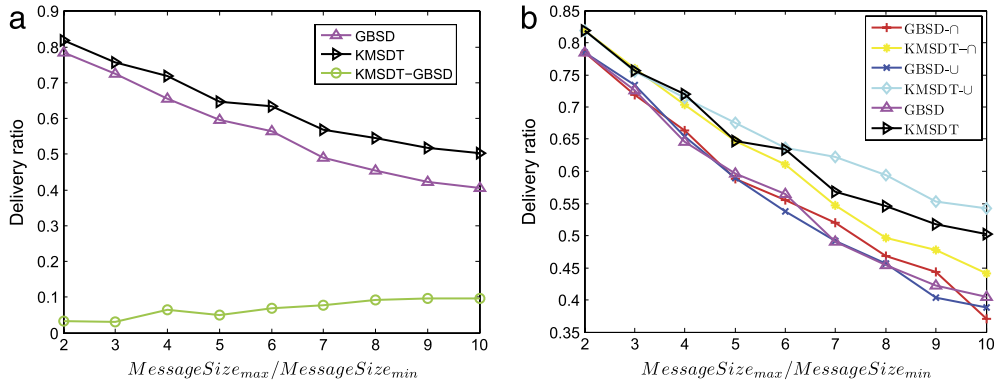


Fig. 12. Delivery ratio under different values of $\frac{Message\ size_{max}}{Message\ size_{min}}$.

Table 3
Different distributions of message sizes.

Message size	Uniform	∩ distribution	∪ distribution
0.5	0.1	$\frac{1}{512}$	$\frac{126}{512}$
1.0	0.1	$\frac{9}{512}$	$\frac{84}{512}$
1.5	0.1	$\frac{36}{512}$	$\frac{36}{512}$
2.0	0.1	$\frac{84}{512}$	$\frac{9}{512}$
2.5	0.1	$\frac{126}{512}$	$\frac{1}{512}$
3.0	0.1	$\frac{126}{512}$	$\frac{1}{512}$
3.5	0.1	$\frac{84}{512}$	$\frac{9}{512}$
4.0	0.1	$\frac{36}{512}$	$\frac{36}{512}$
4.5	0.1	$\frac{9}{512}$	$\frac{84}{512}$
5.0	0.1	$\frac{1}{512}$	$\frac{126}{512}$

decreases along with the increasing abscissa, which results in the remission of congestion. The results show that KMSDT and KMSDP outperform the other buffer-management strategies in terms of the delivery ratio. Fig. 11-(b) indicates that generation rate does not affect average delay to any great extent. The average delay of KMSDP is slightly higher than that of DO, and there is almost no difference between KMSDT and GBSD regarding average delay.

Messages in above experiments are all generated with the sizes arbitrarily chosen from 0.5, 1, 1.5, and 2 MB. Next, in order to verify the applicability of KMSDT, we keep the minimal message size (0.5 MB) unchanged, set the buffer size as 20 MB and vary the ratio between maximal message size and minimal message size from 2 to 10. Results are shown in Fig. 12-(a), which indicates that the delivery ratios of both KMSDT and GBSD decrease along with increase of the ratio between maximal message size and minimal message size. However, KMSDT always performs better than GBSD. At the same time, the trend of difference between them becomes more obvious when the ratio is large enough.

Consider that the message sizes of experiments in Fig. 12-(a) meet the uniform distribution. However, in order to verify that KMSDT still performs well under different distributions of message sizes (as shown in Table 3), the binomial distribution ($p = 0.5$) and corresponding type U distribution are implemented. Results are shown in Fig. 12-(b), which indicates that KMSDT performs better than GBSD no matter in which distribution (∩ means binomial distribution and ∪ means corresponding type ∪ distribution). It is worth noticing that regarding to delivery ratio, KMSDT-∪ > KMSDT > KMSDT-∩, especially when the ratio between maximal message size and minimal message size is large enough. In other words, more disperse the distribution of message sizes is, the better performance KMSDT will get. The more aggregate the distribution of message sizes is, the worse the performance KMSDT will get. It is natural and reasonable, and can also prove that the knapsack-based solution has played a key role in KMSDT.

In conclusion, compared with the other five buffer-management strategies, KMSDT and KMSDP improve the delivery ratio without affecting the overhead ratio or average delay.

5.3.2. Performance evaluation under the real traces: epfl, roma taxi and pmtr

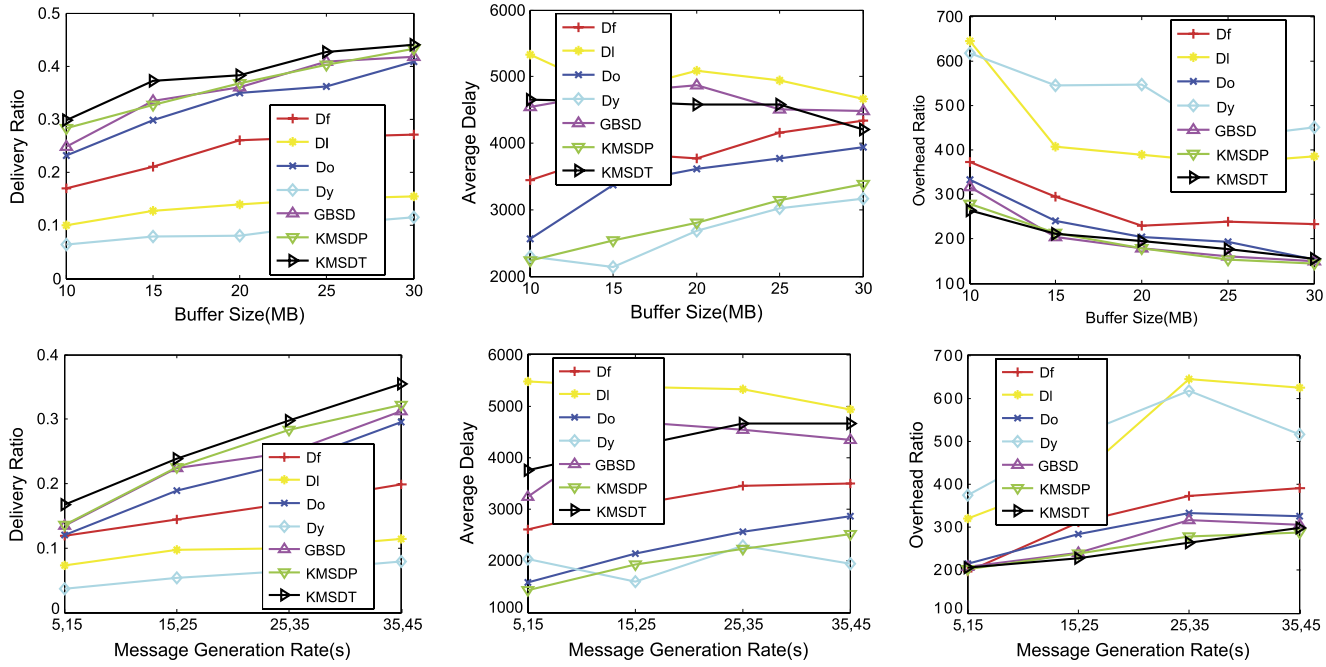
Epfl contains GPS data from 500 San Francisco taxis acquired over 30 days, without loss of generality, we use the data of the first 100 taxis in this paper. For the first part of experiments, we test the epfl dataset using ONE to simulate taxi mobility over the first 18 000 s. Based on the Epidemic routing protocol, seven buffer-management strategies (KMSDT, KMSDP, GBSD, DF, DL, DO, and DY) are implemented. We vary the buffer size and generation rate and observe the variation trends in terms of delivery ratio, average delay, and overhead ratio.

Note that the expectations of the intermeeting time and contact duration (as shown in Fig. 2) under epfl are calculated to be 3287.7 and 105.5 s, respectively. Therefore, we fix the message generation rate to one message per 25–35 s. Through changing the buffer size, we obtain the variation trends of delivery ratio, average delay, and overhead ratio in Fig. 13.

In a realistic network environment, the movement of the taxis lacks the regularity and the nodes cannot communicate with each other as frequently as done in the random-waypoint mobility pattern. As a result, some messages cannot be delivered. Therefore, the delivery ratio obtained in the realistic scenario differs significantly from that obtained from the synthetic random-waypoint scenario. However, according to Fig. 13-(a), we realize that KMSDT and KMSDP achieve the best delivery ratios among the seven buffer-management strategies, and the delivery ratio increases along with the increase of buffer size. Furthermore, Fig. 13-(b) shows that DY obtains the best performance in terms of average delay and KMSDP outperforms the other four buffer-management strategies, although its average delay is higher than that of DY. Moreover, the average delay for KMSDT is similar with that of GBSD. Fig. 13-(c) shows that KMSDT and KMSDP achieve similar overhead ratios, which are better than those of the other five buffer-management strategies.

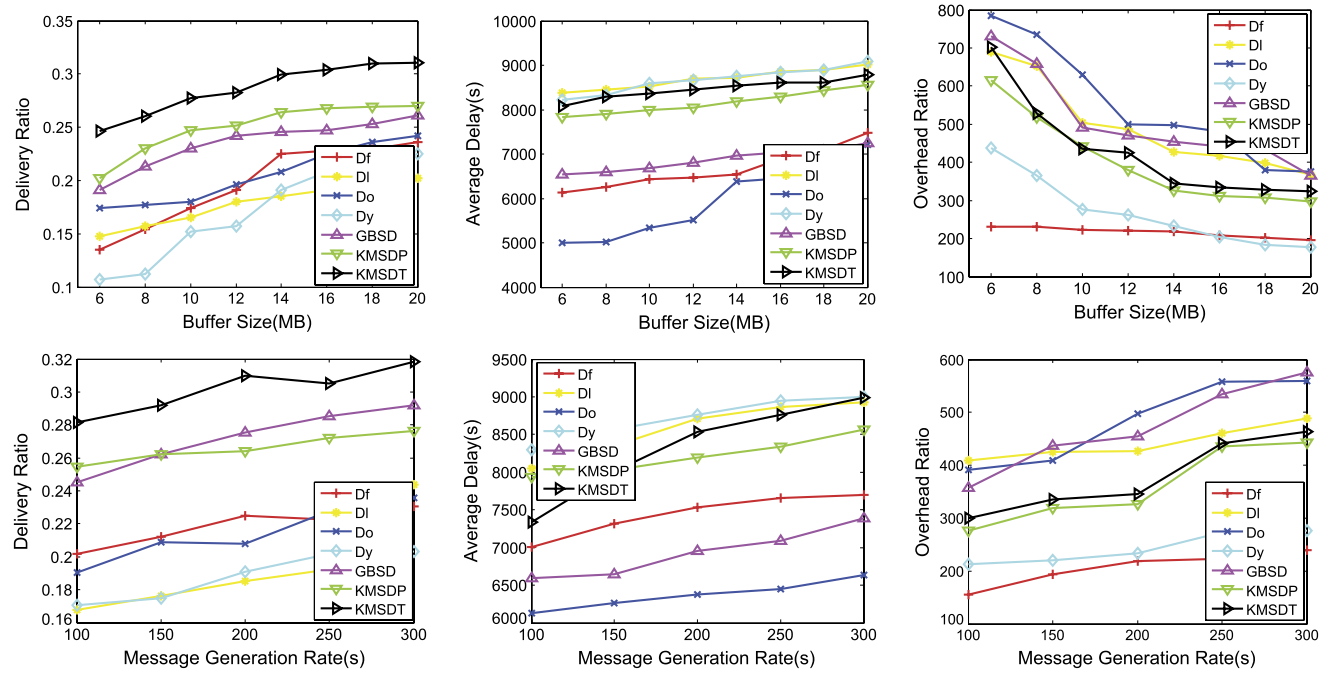
Next, we fix the buffer size to 10 MB and vary the message generation rate. The delivery ratio, average delay, and overhead ratio are plotted in Fig. 13, respectively. Fig. 13-(a) shows that KMSDT achieves the best delivery ratio, which increases linearly with message generation rate. Moreover, KMSDP also obtains a better delivery ratio than GBSD. As shown in Fig. 13-(b), we find that the average delay for KMSDP is much lower than that for GBSD.

In order to further prove the applicability of the proposed KMSDT and KMSDP, we evaluate our strategy under the extra two real-word traces: roma taxi and pmtr. The simulation results are shown in Figs. 14 and 15, respectively. We omit the detailed description of the Figs. 14 and 15, for the reason that the curve-shapes are similar with that of Fig. 13.



(a) Delivery ratio. (b) Average delay. (c) Overhead ratio.

Fig. 13. Delivery ratio, average delay, and overhead ratio as a function of buffer size and message generation rate under the epfl scenario.



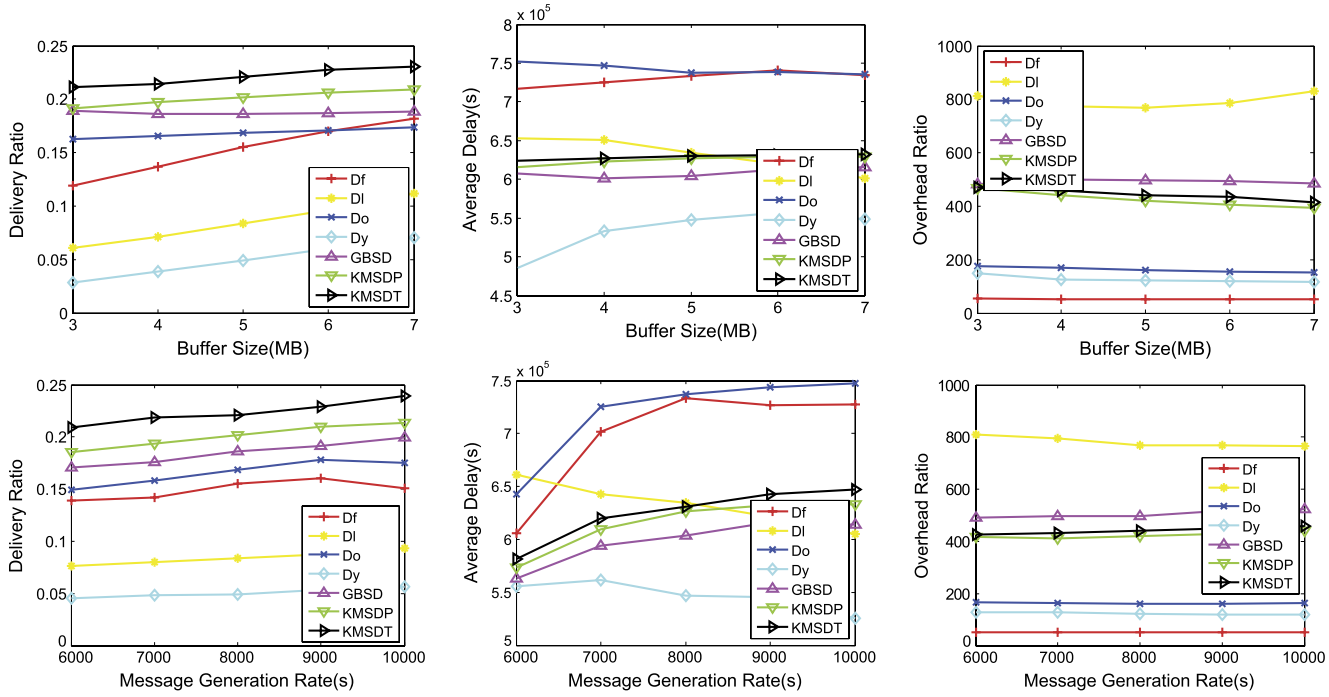
(a) Delivery ratio. (b) Average delay. (c) Overhead ratio.

Fig. 14. Delivery ratio, average delay, and overhead ratio as a function of buffer size and message generation rate under the roma taxi scenario.

To sum up, in realistic scenarios, KMSDT and KMSDP significantly improve the delivery ratio compared with the other five buffer-management strategies (GBSD, DF, DL, DO, and DY), without affecting the average delay.

6. Conclusion

In DTNs, the probabilistic nodal mobility and interruptible wireless links lead to nondeterministic and intermittent connectivity. The store–carry–forward paradigm is used by most routing protocols to efficiently forward messages. However, excessive copies of messages can easily lead to buffer overflowing because of the limited storage space, especially when the bandwidth is also limited and the message sizes differ. In this situation, how to allocate network resources becomes the key point. In this paper, we present an idealized knapsack-based scheduling and drop strategy KMSDT based on the Epidemic routing protocol. In order to improve the delivery ratio, KMSDT schedules messages according to the



(a) Delivery ratio. (b) Average delay. (c) Overhead ratio.

Fig. 15. Delivery ratio, average delay, and overhead ratio as a function of buffer size and message generation rate under the pmtr scenario.

per-unit utility. When the buffer overflows, which messages to drop is decided according to the solution of the knapsack problem. However, KMSDT cannot be applied in the realistic network environment because it requires global parameters. Therefore, we develop a practical scheduling and drop strategy KMSDP. KMSDP uses the distributed collected history information to approximate the global information, and uses these estimated parameters to calculate the utility. We conducted simulations in ONE under both the synthetic random-waypoint mobility scenario and the real traces: epfl, roma taxi and pmtr. The simulation results show that, compared with other buffer-management strategies, KMSDT and KMSDP significantly improve the delivery ratio without affecting the average delay. KMSDT and KMSDP aim to maximize the delivery ratio without considering other performance metrics. Future work will focus on developing a more efficient scheduling and drop strategy to optimize both the delivery ratio and the average delay.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 61272412, Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20120061110044, Jilin Province Science and Technology Development Program under Grant No. 20120303, and China Scholarship Council (No. [2014]3026). This work is supported in part by NSF grants CNS 149860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, ECCS 1231461, ECCS 1128209, and CNS 1138963.

References

[1] I. Akyildiz, B. Akan, C. Chen, InterPlaNetary Internet: state-of-the-art and research challenges, Comput. Netw. 43 (2) (2003) 75–112.
[2] B. Aruna, L. Brian, V. Arun, DTN routing as a resource allocation problem, in: Proc. of ACM SIGCOMM 2007, pp. 373–384.
[3] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, A. Rabuffi, CRAWDAD data set roma/taxi (v. 2014-07-17), July 2014. Downloaded from http://crawdad.org/roma/taxi/.
[4] S. Burleigh, A. Hooke, L. Torgerson, Delay-tolerant networking: an approach to interplanetary Internet, IEEE Commun. Mag. 41 (6) (2003) 128–136.
[5] J. Cao, Y. Zhang, L. Xie, Consistency of cooperative caching in mobile peer-to-peer systems over MANET, Int. J. Parallel Emergent Distrib. Syst. 21 (3) (2006) 151–168.
[6] K. Dohyung, P. Hanjin, Y. Ikjun, Minimizing the impact of buffer overflow in DTNs, in: Proc. of International Conference on Future Internet Technologies, CFI, 2008.
[7] A. Elwhishi, P. Ho, K. Naik, B. Shihada, A novel message scheduling framework for delay tolerant networks routing, IEEE Trans. Parallel Distrib. Syst. 24 (5) (2013) 871–880.
[8] V. Erramilli, M. Crovella, Forwarding in opportunistic networks with resource constraints, in: Proceedings of the Third ACM Workshop on Challenged Networks, in: Proc. of ACM CHANTS 2008, pp. 41–48.
[9] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proc. of ACM SIGCOMM 2003, pp. 27–34.
[10] W. Gao, Q. Li, G. Cao, Forwarding redundancy in opportunistic mobile networks: Investigation and elimination, in: Proc. of IEEE INFOCOM 2014, pp. 1–9.
[11] http://en.wikipedia.org/wiki/Wiki/Nyquistrem.
[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet, in: Proc. of ASPLOS 2002, pp. 96–107.
[13] A. Krifa, C. Barakat, An optimal joint scheduling and drop policy for delay tolerant networks, in: Proc. of IEEE WoWMoM 2008, pp. 1–6.
[14] A. Krifa, C. Barakat, Optimal buffer management policies for delay tolerant networks, in: Proc. of IEEE SECON 2008, pp. 260–268.
[15] A. Krifa, C. Barakat, Message drop and scheduling in DTNs: Theory and practice, IEEE Trans. Mob. Comput. 11 (9) (2012) 1470–1483.
[16] R. Krishnan, P. Basu, J. Mikkelson, The spindle disruption-tolerant networking system, in: Proc. of MILCOM 2007, pp. 1–7.
[17] Q. Li, S. Zhu, G. Cao, Routing in socially selfish delay tolerant networks, in: Proc. of IEEE INFOCOM 2010, pp. 2301–2309.
[18] A. Lindgren, K.S. Phanse, Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks, in: Proc. of IEEE Comsware 2006, pp. 1–10.
[19] P. Meroni, S. Gaito, E. Pagani, G.P. Rossi, CRAWDAD data set unimi/pmtr (v. 2008-12-01), December 2008. Downloaded from http://crawdad.org/unimi/pmtr/.
[20] P. Pawelczak, P.R. Venkatesha, L. Xia, Cognitive radio emergency networks - requirements and design, in: Proc. of IEEE DySPAN 2005, pp. 601–606.
[21] A. Pentland, R. Fletcher, A. Hasson, Daknet: rethinking connectivity in developing nations, IEEE Comput. 37 (1) (2004) 78–83.
[22] M. Piorkowski, N. Sarafijanovic-Djukic, M. Grossglauser, CRAWDAD data set epfl/mobility (v. 2009-02-24), February 2009. Downloaded from http://crawdad.org/epfl/mobility/.
[23] G. Robin, N. Philippe, K. Ger, Message delay in manet, in: Proc. of ACM SIGMETRICS 2005, pp. 412–413.

[24] L. Sassatelli, A. Ali, M. Panda, T. Chahed, E. Altman, Reliable transport in delay-tolerant networks with opportunistic routing, IEEE Trans. Wireless Commun. 13 (10) (2014) 5546–5557.
[25] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Performance analysis of mobility-assisted routing, in: Proc. of ACM Mobihoc 2006, pp. 49–60.
[26] M.Y.S. Uddin, H. Ahmadi, T. Abdelzaher, R. Kravets, Intercontact routing for energy constrained disaster response networks, IEEE Trans. Mob. Comput. 12 (10) (2013) 1986–1998.
[27] A. Vahdat, D. Becker, Epidemic routing for partially-connected ad hoc networks, Tech. rep., Apirl 2000.
[28] R. Wahidabanu, G. Fathima, A new queuing policy for delay tolerant networks, Int. J. Comput. Appl. 1 (20) (2010) 56–60.
[29] E. Wang, Y. Yang, J. Wu, A knapsack-based message scheduling and drop strategy for delay-tolerant networks, in: Proc. of the 12th European International Conference on Wireless Sensor Networks, EWSN 2015, pp. 120–134.
[30] J. Wu, Y. Wang, Hypercube-based multi-path social feature routing in human contact networks, IEEE Trans. Comput. 63 (2) (2014) 383–396.
[31] M. Xiao, J. Wu, L. Huang, Community-aware opportunistic routing in mobile social networks [supplemental], IEEE Trans. Comput. 63 (7) (2014) 1682–1695.
[32] L. Yong, J.Q. Meng, Adaptive optimal buffer management policies for realistic DTNs, in: Proc. of IEEE GLOBECOM 2009, pp. 1–5.
[33] Z. Zhang, Z. Xianwei, G. Jiwen, Noncooperative dynamic routing with bandwidth constraint in intermittently connected deep space information networks under scheduled contacts, Wirel. Pers. Commun. 68 (4) (2013) 1255–1285.
[34] H. Zheng, Y. Wang, J. Wu, Optimizing multi-copy two-hop routing in mobile social networks, in: Proc. of IEEE SECON 2014, June 2014.

En Wang received his B.E. degree in Software Engineering from Jilin University, Changchun, Jilin, China, in 2011; and M.E. degree in Computer Science and Technology from Jilin University, Changchun, Jilin, China, in 2013. He is currently a Ph.D. candidate in the Department of Computer Science and Technology, Jilin University, Changchun, Jilin. And he is also a visiting scholar in the Department of Computer and Information Sciences, Temple University, Philadelphia, PA. His research interests are in the design and realize a variety of novel routing protocols and buffer-management strategies in delay tolerant networks. He has authored 8 papers on delay tolerant networks and social networks. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices.

Yongjian Yang received his B.E. degree in Automatization from Jilin University of Technology, Changchun, Jilin, China, in 1983; and M.E. degree in Computer Communication from Beijing University of Post and Telecommunications, Beijing, China, in 1991; and his Ph.D. in Software and theory of Computer from Jilin University, Changchun, Jilin, China, in 2005. He is currently a professor and a Ph.D. supervisor at Jilin University, the Vice Dean of Software College of Jilin University, also Director of Key lab under the Ministry of Information Industry, Standing Director of Communication Academy, member of the Computer Science Academy of Jilin Province. His research interests include: Theory and software technology of network intelligence management; Key technology research of wireless mobile communication and services; research and exploiture for next generation services foundation and key productions on wireless mobile communication. He participated 3 projects of NSFC, 863 and funded by National Education Ministry for Doctoral Base Foundation. He has charged 12 projects of NSFC, key projects of Ministry of Information Industry, Middle and Young Science and Technology Developing Funds, Jilin provincial programs, ShenZhen, ZhuHai and Changchun. As the 1st author, he has published more than 60 papers in national and foreign journals.

Jie Wu received his B.S. in computer engineering and M.S. in computer science from Shanghai University of Science and Technology (now Shanghai University), Shanghai, China, in 1982 and 1985, respectively, and his Ph.D. in computer engineering from Florida Atlantic University, Boca Raton, in 1989. Jie Wu is the chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University. Prior to joining Temple University, he was a program director at the National Science Foundation and Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Computers, IEEE Transactions on Service Computing, and Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006 and IEEE IPDPS 2008 and program co-chair for IEEE INFOCOM 2011. Currently, he is serving as general chair for IEEE ICDCS 2013 and ACM MobiHoc 2014, and program chair for CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.