




DTN-Balance: A Forwarding-Capacity and Forwarding-Queue Aware Routing for Self-organizing DTNs

Weitao Wang¹ · Yuebin Bai¹  · Peng Feng¹ · Jun Huang² · Mo Sha³ · Jianpei Tantai¹

Accepted: 23 December 2020 / Published online: 13 February 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC part of Springer Nature 2021

Abstract

In delay-tolerant networks (DTNs), intermittent network connectivity and lack of global system information pose serious challenges to achieve effective data forwarding. Most state-of-the-art DTN routing algorithms are based on hill-climbing heuristics in order to select the best available next hop to achieve satisfactory network throughput and routing efficiency. An adverse consequence of this approach is that a small subset of good users take on most of the forwarding tasks. This can quickly deplete scarce resources (e.g. storage, battery, etc.) in heavily utilized devices which degrades the network reliability. A system with a significant amount of traffic carried by a small number of users is not robust to denial of service attacks and random failures. To overcome these deficiencies, this paper proposes a new routing algorithm, DTN-Balance, that takes the forwarding capacity and forwarding queue of the relay nodes into account to achieve a better load distribution in the network. For this, we defined a new routing metric called message forwarding utility combining nodal available bandwidth and forwarding workload. Applying small world theory, we impose an upper bound on the end-to-end hop count that results in a sharp increase in routing efficiency. Queued messages in a forwarding node are arranged by DTN-Balance based on message dropping utility metric for a more intelligent decision in the case of a message drop. The performance of our method is compared with that of the existing algorithms by simulations on real DTN traces. The results show that our algorithm provides outstanding forward efficiency at the expense of a small drop in the throughput.

Keywords DTN · Routing · Virtual bandwidth · Load balance · Efficiency forwarding · Queue control

1 Introduction

In delay-tolerant networks (DTNs) [1,2], due to low node density and mobility, mobile devices such as smartphones, laptops and PDAs are only intermittently connected. Due to frequent network partitions in DTNs, the messages are usually transferred in a store-carry-forward manner [3] that results in high end-to-end latency. However, DTNs could contribute

Extended author information available on the last page of the article

to significant reduction in infrastructure costs and may result in increasing user bandwidth by orders of magnitude.

An efficient routing and forwarding mechanism is essential for the success of DTNs. A routing mechanism based on store-carry-forward manner in DTNs is responsible for calculating the best next hop for forwarding a message in an efficient manner possibly taking network wide considerations such as traffic load distribution into account. Except for [4–9], most of the existing works are aimed at maximizing throughput by studying how to maximize the delivery ratio and minimize the number of forwarding, thus there has been no attention to scalability and reliability. Some special characteristics of DTN, such as limited network coverage, device outages, roaming and human interactions make these networks prone to failures. There are many reasons for failure: (1) Devices cannot manage the load caused by insufficient resources to fail [10], (2) Central nodes with high forwarding contributions may be targeted by denial of service attacks [11], and (3) Many DTN nodes, especially bottleneck nodes, are reluctant to join the network due to mismatch between the tasks undertaken (the total number of messages forwarded) and the benefits received (the number of self owned messages forwarded) [12]. Such failure circumstances result in scalability and reliability reduction that illustrates the need for effective forwarding and workload balancing techniques in DTNs.

Reducing forward cost and improving forward efficiency simultaneously remains to be an important problem in the DTNs and efficient solutions are essential for their successful deployment.

Node connectivity pattern in DTN networks have statistical properties similar to social networks. The fat-tail connectivity distribution is one of the characteristics of DTN, which means that most nodes have only a few connectivity, and multi-connectivity nodes only account for a small part. Inevitably, nodes with high connectivity carry majority of the traffic. However, these nodes are usually resource limited (e.g. in terms of buffer size); hence, this results in an unfair load distribution and low forwarding efficiency.

Since irregular network structure can cause load imbalance, a natural solution is to upgrade all bottleneck nodes. But every single node has limited resources (i.e., battery life) and different management domain (i.e., individual user), thus it cannot be compensated. These routing algorithms based on best-next-hop doctrine and hill-climbing heuristics do not consider this significant workload imbalance either. This results in decline of network survivability as major active nodes are bound to run out quickly. Moreover, even before nodes failure, node's buffer is likely to overflow leading to heavy packet loss.

To deal with this problem, we propose the DTN-Balance algorithm that fully utilizes all nodes' forwarding capacities and workloads to guide the forwarding decisions. DTN-Balance considers the change of forwarding capabilities and workloads of the neighboring nodes and aims to create a balance among forwarding cost, load distribution and performance, and distribute the traffic evenly among its neighbors. Compared with the current classic routing algorithms, DTN-Balance algorithm can achieve higher efficiency and better load balance, that is demonstrated by simulation results using realistic traces (UIM [40] and Infocom2006 [41]).

The rest of the paper is organized as follows. Section 2 presents the recent related works. Section 3 introduces the concept of "forwarding capacity" and elaborates on the proposed queue control mechanism. Section 4 introduces the proposed routing algorithm in detail. Section 5 describes performance evaluation metrics and simulation results. Finally, Sect. 6 concludes this paper and presents future research directions.

2 Related Works

Due to enormous potential benefits of DTNs, a lot of work has been done in this field, focusing on routing algorithm. Initial data forwarding techniques in DTNs originated from Epidemic Routing [13], the algorithm replicates and transmits messages by flooding to nodes that have not cached the messages, so as to achieve the goal of maximizing network throughput (i.e., the amount of data transmitted from source nodes to destination nodes in unit time). Epidemic routing maximizes throughput at the cost of flooding, thus its efficiency is very low, because each message is copied and forwarded without limit until the network is flooded by the copies of the message. In view of the shortcomings of Epidemic routing, the following researches approach the performance of Epidemic routing by reducing the number of message copies created in the network. At present, the number of copies is not considered in most forwarding schemes. Only Spray-and-Focus scheme [14], the conservative forwarding scheme in [15] and the scheme of [16] as well as [17] consider a fixed number of copies scenario, single copy scenario and the limited number of copies scenario, respectively. Most of the current forwarding schemes are heuristic algorithms based on finding the optimal next hop, which forwards messages to nodes with the maximum probability of delivering messages to destination nodes. There are two forms of this method: one is that a relay node forwards data to its neighbor node with the optimal value of metric, such as Delegation Forwarding [18]; the other is that all neighbor nodes with their value of metric better than that of current node act as next hop node of forwarding data, such as Compare-and-Forward [19].

This metric mentioned above is called data forwarding metric, which is used to measure the ability of a node to forward data to the destination node. This metric is mainly used in heuristic forwarding schemes based on finding the optimal next hop. The same forwarding scheme can use different forwarding metrics to meet different performance requirements. The forwarding metrics can be simply divided into two types: one is related to data destination; the other is irrelevant to data destination. In the first type, the probability of meeting between a node and message destination node is taken as the metric, such as [20–28,37–39]. Due to the lack of global information, the ability to connect other nodes is taken by the second type as the metric, such as [29–36]. In the second type schemes, data is forwarded based on social contact patterns. As an example, DTN-FLOW [34] builds an overlay of landmarks on top of a DTN and uses the landmarks overlay to route the messages by computing the virtual bandwidth between landmarks.

In summary, except [4–9], none of the other routing methods mentioned above consider network reliability and survivability. On the contrary, DTN-Balance predicts node's forwarding capacity by measuring the virtual bandwidth and creates a balance among load distribution, forwarding cost and performance to improve network reliability and survivability.

3 Data Forwarding Capacity Metric

In this section, we introduce the data forwarding metric for a mobile node based on that node's bandwidth to others in the network. We focus on measuring available virtual bandwidth within the given scope in terms of the average number of transmitted bytes per unit time.

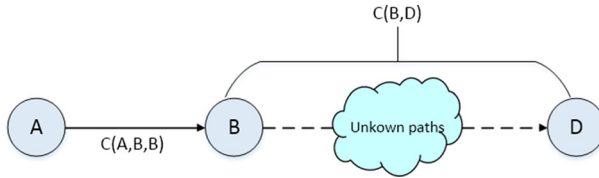


Fig. 1 A “virtual path” consisting of node A (source), B (next hop) and D (destination)

3.1 Virtual Bandwidth

In wired or wireless networks with end-to-end connection, usually “volume” or “bandwidth” is used to represent the transmission capacity of the link. In this paper, in the absence of predefined end-to-end path, a link is considered to be a single-hop path to a neighboring node. Since the available bandwidth to a neighboring node does not necessarily represent the end-to-end service capacity, we use the term “virtual bandwidth” to represent the transmission capacity of a network segment or a single-hop path in DTNs. To be precise, we define the virtual bandwidth as follows.

Definition 1 Virtual Bandwidth: is defined as the average number of bytes that can be transferred from node X to node Y per unit of time, and is denoted by $C(X, Y)$.

Virtual bandwidth is a statistical (and not a deterministic) quantity. It is also directional, i.e., the virtual bandwidth from node X to node Y is different with the virtual bandwidth from node Y to X . Let $C(X, Y)$ denotes the virtual bandwidth from X to Y , then $C(X, Y)$ is usually different from $C(Y, X)$. In the definition of virtual bandwidth, node X and Y does not need to be neighbors; hence, the data from node X to Y is transferred directly or indirectly through other nodes.

3.2 Virtual Path

In DTNs, when relaying a message, the sending node chooses a next-hop node that can transmit packets to the destination. In choosing the next-hop node, it is immaterial how that node transfers the message to the destination. We only care about the next-hop nodes’ likely transfer rate to the destination. Thus we can treat the path consisting of source, next hop and destination as a “virtual path”, depicted in Fig. 1. Here, we call it a “virtual path” since there is no fixed end-to-end path in DTNs as the connection to neighboring nodes could break frequently and also we do not care and may even not know how next hop node reaches the destination. Thus, we use a cloud to represent the connection between the next hop and the destination. To be precise, the virtual path is defined as in the following.

Definition 2 Virtual Path: A virtual path is denoted by $vl(X, Y, Z)$ where X is the source node, Y is the second hop node and Z is the destination.

A virtual path $vl(X, Y, Z)$ which starts at node X and reaches node Z via next-hop node Y exists if and only if node X can directly reach Y and node Y can directly or indirectly reach Z . Note, existence of $vl(X, Y, Z)$ does not guarantee existence of $vl(Z, Y, X)$. The virtual bandwidth of $vl(X, Y, Z)$ is represented by $C(X, Y, Z)$, measuring the average bytes per unit time transferred from node X to Z via node Y . We know $C(X, Y, Z) \neq C(Z, Y, X)$.

3.3 Virtual Bandwidth Calculation

In this subsection, we describe an approach for calculation of the virtual bandwidth. Let's first consider the virtual path $vl(A, B, D)$ in Fig. 1 and let's assume we know the virtual bandwidth $C(A, B, B)$ of $vl(A, B, B)$ and the virtual bandwidth $C(B, D)$ from node B to D . Consider one-byte message transferred from node A to D via this virtual path. The time it takes to relay this message from node A to B is $\frac{1}{C(A,B,B)}$. Once node B receives this message, it delivers this message to D and the time that it takes is $\frac{1}{C(B,D)}$. Then this one byte message, the time taken to reach node D from source A via B is $\frac{1}{C(A,B,B)} + \frac{1}{C(B,D)}$. Thus, the virtual bandwidth $C(A, B, D)$ of $vl(A, B, D)$ is

$$\begin{aligned}
 C(A, B, D) &= \frac{1}{\frac{1}{C(A,B,B)} + \frac{1}{C(B,D)}} \\
 &= \frac{C(A,B,B) \times C(B,D)}{C(A,B,B) + C(B,D)}
 \end{aligned}
 \tag{1}$$

We now consider calculating $C(X, Y)$ where there could be multiple virtual paths between X and Y . Let us consider the case where there exist two virtual paths as shown in Fig. 2. The two virtual paths from node A to D are $vl(A, B, D)$ and $vl(A, C, D)$. We consider sending a one byte message from node A to D . If the message is transferred along $vl(A, B, D)$, the transit time is $\frac{1}{C(A,B,D)}$. Similarly, the time taken along $vl(A, C, D)$ is $\frac{1}{C(A,C,D)}$. In each transmission event, one path or the other may be used. As in regular networks, when transferring a message, we prefer paths with higher local virtual bandwidth. The local virtual bandwidth in the two virtual paths from A to D are $C(A, B, B)$ and $C(A, C, C)$ respectively. Thus we let $\frac{C(A,B,B)}{C(A,B,B)+C(A,C,C)}$ and $\frac{C(A,C,C)}{C(A,B,B)+C(A,C,C)}$ denote the weights attached to those two virtual paths. Then, the average time taken from node A to D ($T(A, D)$) is:

$$\begin{aligned}
 T(A, D) &= \frac{C(A,B,B)}{C(A,B,B)+C(A,C,C)} \times \frac{1}{C(A,B,D)} \\
 &\quad + \frac{C(A,C,C)}{C(A,B,B)+C(A,C,C)} \times \frac{1}{C(A,C,D)}
 \end{aligned}
 \tag{2}$$

and the virtual bandwidth from node A to D is

$$C(A, D) = \frac{1}{T(A, D)}
 \tag{3}$$

To be general, let node X can reach node Z via n virtual paths and the i_{th} virtual path denoted by $vl(X, Y_i, Z)$. Then the virtual bandwidth from node X to Z is defined as follows.

Definition 3 Total Virtual Bandwidth: The total virtual bandwidth between node X and Z with n existing virtual paths is

$$\begin{aligned}
 C(X, Z) &= \frac{1}{\sum_{i=1}^n \frac{C(X,Y_i,Y_i)}{\sum_{j=1}^n C(X,Y_j,Y_j)} \times \frac{1}{C(X,Y_i,Z)}} \\
 &= \frac{\sum_{j=1}^n C(X, Y_j, Y_j)}{n + \sum_{i=1}^n \frac{C(X,Y_i,Y_i)}{C(Y_i,Z)}}
 \end{aligned}$$

Fig. 2 Two virtual paths from node A to D

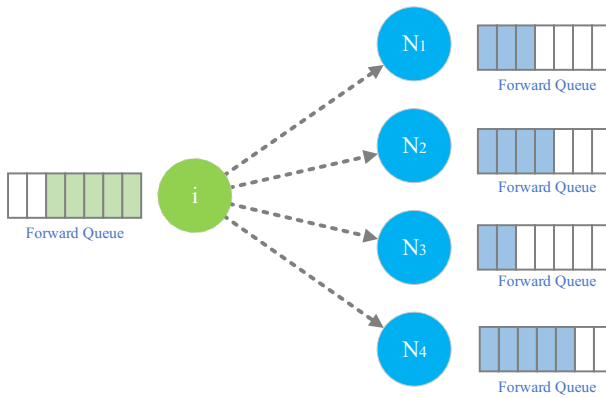
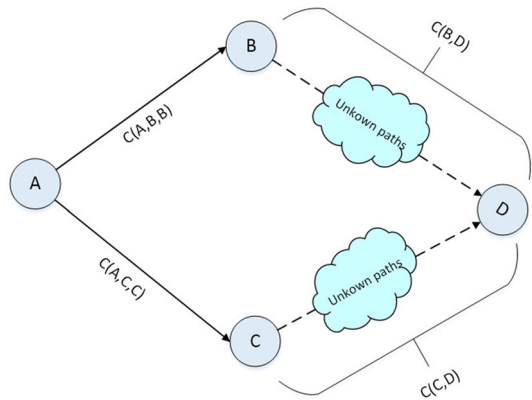


Fig. 3 The data packet transmission process in DTN

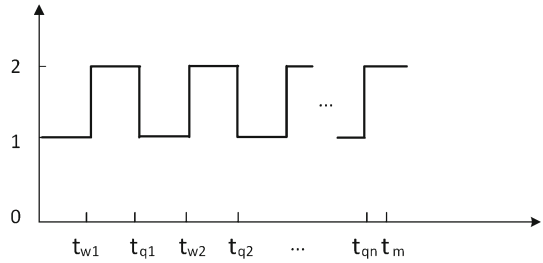
3.4 Virtual Bandwidth Calculation for Direct Link

Referring to to Definition 1, the virtual bandwidth $C(A, B, B)$ of a path $vl(A, B, B)$ is equal to the amount of data transferred from node A to its neighbor node B divided by the time spent for the transfer of this data, $et_{vl(A, B, B)}$. When the network traffic is low, the queuing delay can be ignored and hence $et_{vl(A, B, B)}$ is determined by link down times. However, the queuing delay cannot be ignored under high network traffic. If the link is regarded as a service window, and if we consider link down times as vacations, then packet transmission process on a link can be modeled as a queuing system with vacations. In DTN, one node may connect through multiple nodes, as is shown in Fig. 3. Thus the data packet transmission process can be modeled as multi-server queuing system with vacations.

We consider a message m in the message forwarding queue to be forwarded out from node i . Let t_w be the waiting time until at least one link is up. Message m should wait until the prior messages in the queue have been served. We denote the queuing time by t_q , the message transfer time by t_m , and link up time by t_{up} .

Every up link is considered as a service window. If $t_q > t_w + t_{up}$, the message m cannot be transferred in the current service window and we must wait for the next transfer opportunity (window). Assume m needs to wait n rounds. Each round consists of a wait time for link to be up and a wait time for prior messages to be forwarded (queuing time). In Fig. 4, “1”

Fig. 4 The compositions of one-hop delay in DTN



represents the wait state for a link to be up, and “2” represents the wait state for prior messages to be forwarded. Then the total forwarding delay of message m from node i to node j is

$$d_m(i, j) = \sum_{k=1}^n (t_{w_k} + t_{q_k}) + t_m \tag{4}$$

We make the following assumptions for our multiple windows queuing system with vacations:

1. The packet arrival process is assumed to be Poisson with intensity λ ;
2. The message transmission times are assumed to be exponentially distributed with rate μ_i for each link l_i ;
3. The buffer size is VN .

Let us have c direct links as service windows, we define ρ_c as traffic intensity where $\rho_c = \frac{\lambda}{\sum_{i=1}^c \mu_i}$.

If $0 < \rho_c < 1$, according to our assumptions above, the system is an M/M/C queuing model with random vacation. If $\rho_c \geq 1$, the system will not be stable which means that packets will start to drop due to buffer overflow.

When the buffer is not full, new arrivals are appended to the end of the queue. In case a new arrival encounters a full buffer, it competes with previous packets in the buffer, and then it may be dropped or is appended to the end of the queue. We discuss the message queuing delay in these two cases in the following.

3.4.1 Stable State

In this case, we have $0 < \lambda < c\bar{\mu}$ where $\bar{\mu}$ is the average service rate, and $\bar{\mu} = \frac{\sum_{i=1}^c \mu_i}{c}$. We define $N(t)$ as the number of packets in the system at time t . Under the assumptions previously discussed, $N(t)$ is a Birth-Death Process. As such, according to Little’s Formula, the message average wait time in the classic queuing system is:

$$W_q = \left(c\rho_c + \frac{\rho_c P_\infty}{1 - \rho_c} \right) / \lambda - \frac{1}{\bar{\mu}} \tag{5}$$

where P_∞ is the probability that all links are busy (i.e. up state), $\bar{\mu}$ is the average service rate, and P_∞ can be predicted as following:

$$P_\infty = \prod_{i=1}^c P_{busy}^i \tag{6}$$

where P_{busy}^i can be calculated by busy and leisure statistics.

Let $Q = \{l_1^i, l_2^i, \dots, l_T^i\}$ be the perceived link down state vector, and l_j^i be the down-time at state j . Then the average down time \tilde{l} is calculated as:

$$\tilde{l} = \frac{\sum_{j=1}^T l_j^i}{T} \tag{7}$$

Then the average vacation period is:

$$\tilde{l} = \frac{\sum_{i=1}^C \tilde{l}^i}{C} \tag{8}$$

Assume s_{ij} is the transmit speed of link ij , then

$$t_m = \frac{Size(m)}{s_{ij}} \tag{9}$$

Thus the average delay is obtained as:

$$\tilde{d}^1 = (\lfloor W_q / \tilde{l} \rfloor) \cdot \tilde{l} + W_q + \frac{size(m)}{s_{ij}} \tag{10}$$

where $Size(m)$ is message size of m .

3.4.2 Unstable State

In this condition, $\lambda > c\bar{\mu}$, the buffer is expected to be fully filled at most of the time. In order to simplify model, we consider the queuing delay as:

$$W_q = \frac{VN}{\mu} \tag{11}$$

Thus the average delay is

$$\tilde{d}^2 = (\lfloor W_q / \tilde{l} \rfloor) \cdot \tilde{l} + W_q + \frac{1}{s_{ij}} \tag{12}$$

Thus the virtual bandwidth for direct link can be calculated as follows:

$$C(A, B, B) = \begin{cases} \tilde{l} + \frac{1}{s_{ij}}, & \rho_c = 0 \\ \frac{1}{\tilde{d}^1}, & 0 < \rho_c < 1 \\ \frac{1}{\tilde{d}^2}, & \rho_c \geq 1 \end{cases} . \tag{13}$$

4 The Proposed Routing Algorithm

With the introduction of virtual link in Sect. 3, in this section, we develop our message forwarding schedule to choose the next hop, namely DTN-Balance. To achieve satisfactory load distribution, high routing efficiency and acceptable network throughput, we use forwarding capacity information of the neighboring nodes as well as the status of their forwarding queues. In the following subsections, we first introduce the data structures used in our algorithm. We then describe the data update and recalculation method before introducing the proposed message priority queue control and forwarding schedule.

4.1 Data Structures

To calculate the virtual bandwidth calculation method in Theorem 3, we need two pieces of information: $C(X, Y_i, Y_i)$ and $C(Y_i, Z)$. We thus need to store these two kinds of forwarding capacities. We, therefore, create two tables at each node, i.e. Virtual Node Bandwidth Table (VNBT) and Virtual Path Bandwidth Table (VPBT). The structures of the two tables are illustrated in Tables 1 and 2. Table 1 contains the destination node Z_i and the virtual path's virtual bandwidth from current node X to destination Z_i . In Table 2, the *virtual bandwidth* is the one-hop virtual path's virtual bandwidth. $VPBT_X$ contains all $C(X, Y_i, Y_i)$, where Y_i is the node that X can directly deliver message to. The last item in this table, $VNBT_Y$, is a pointer to the neighbour node Y 's $VNBT$ that contains all $C(Y, Z_i)$, where Z_i is the node that Y can deliver message to (directly or not). We can see that we have a nested data structure where every row of $VPBT$ table points to a $VNBT$. Later, we will see that this design facilitates the packet forwarding schedule calculation.

One should note that the records in $VNBT$ are calculated based on the records in $VPBT$ and vice versa. Initially, both $NVBT_X$ and $VPBT_X$ are empty. Once node X meets node Y_i , if there is no record for node Y_i , X creates a record for Y_i . Otherwise, updates the record. When $VPBT_X$ is not empty, we can use $VPBT_X$ to calculate virtual bandwidth and put the result into $VNBT_X$. The updated $VNBT_X$ will then be used in other $VPBT$ s. This recursive operations continue until all tables converge to their final values.

Table 1 Node virtual bandwidth table (NVBT)

Node	Virtual bandwidth
Z_1	$C(X, Z_1)$
...	...
Z_n	$C(X, Z_n)$

Table 2 Virtual path bandwidth table (VPBT)

Node	Virtual bandwidth	$NVBT_Y$
Y_1	$C(X, Y_1, Y_1)$	$NVBT_{Y_1}$
...
Y_m	$C(X, Y_m, Y_m)$	$NVBT_{Y_m}$

Algorithm 1 Node X 's behavior when it meet Y_i **Input:**

the encountered node, Y_i
 transmit speed between X and Y_i , sp

Output:

the updated $VPBT_X$ and $NVBT_X$
 1: notify node Y_i recalculate whole $NVBT_{Y_i}$
 2: request $NVBT_{Y_i}$ from node Y_i
 3: **if** $C(X, Y_i, Y_i)$ does not exist in $VPBT_X$ **then**
 4: create a new record for Y_i in $VPBT_X$
 5: set $C(X, Y_i, Y_i) := 0$
 6: set $NVBT_{Y_i} := \text{empty table}$
 7: **end if**
 8: update $NVBT_X$, recalculate $C(X, Y_i)$ using *Algorithm 3*
 9: **for** every record $C(Y_i, Z_k)$ in $NVBT_{Y_i}$ **do**
 10: **if** $NVBT_{Y_i}$ of $VPBT_X$ does not have $C(Y_i, Z_k)$ **then**
 11: add $C(Y_i, Z_k)$ to $NVBT_{Y_i}$ of $VPBT_X$
 12: **else**
 13: update the corresponding record
 14: **end if**
 15: recalculate $\widehat{C}(X, Z_k)$ using *Algorithm 4*
 16: update $NVBT_X$, set $C(X, Z_k) := \widehat{C}(X, Z_k)$
 17: **end for**

Algorithm 2 Node X 's behavior when it separate from Y_i **Input:**

the separated node, Y_i
 transmit speed between X and Y_i , sp_i
 perceived link down states list between X and Y_i , Q

Output:

the updated $VPBT_X$ and $NVBT_X$
 1: **if** a record entry contains the Y_i in $VPBT_X$ **then**
 2: update $C(X, Y_i, Y_i)$ using *Algorithm 3*
 3: update $NVBT_X$, set $C(X, Y_i) := C(X, Y_i, Y_i)$
 4: **for** every record $C(Y_i, Z_k)$ in $NVBT_{Y_i}$ of $VPBT_{Y_i}$ **do**
 5: recalculate $\widehat{C}(X, Z_k)$ using *Algorithm 4*
 6: update $NVBT_X$ set, $C(X, Z_k) := \widehat{C}(X, Z_k)$
 7: **end for**
 8: **end if**

4.2 Spread and Recalculation

In order to calculate virtual bandwidth, nodes need to gather virtual bandwidth information of the network and store these information into their $VNBTs$ and $VPBTs$. When two nodes meet (their link is up), the first thing they need to do is to exchange their $VNBTs$. For example, node X meets Y_i , X and Y_i both exchange their $VNBTs$ with each other in order to update their $VPBTs$. On the other hand, when both $VPBTs$ are updated, $VNBTs$ of both nodes also need to be updated as both X and Y_i recalculate virtual bandwidth for each item of their $VNBTs$, respectively. The update process when node X meets node Y_i is formally described in Algorithm 1.

In DTNs, connections with other nodes break frequently. Upon a break, the node needs to recalculate its virtual bandwidth tables. Take node X and Y_i for an instance, when node X separates with Y_i , the virtual bandwidth from X to Y_i changes, thus X also needs to update $C(X, Y_i, Y_i)$. To update records in both tables, each node needs to refresh $C(X, Y_i, Y_i)$ and

Algorithm 3 Calculate virtual bandwidth of direct link**Input:**

perceived link down states list between X and Y_i , Q
 arriving ratio of messages to be forwarded per minute, λ
 transmit speed between X and current each neighbour Y_i , sp_i , $i = 1, \dots, C$

Output:

virtual bandwidth for direct link, $C(X, Y_i, Y_i)$
 1: calculate each link service ratio $\mu_i = P_{busy}^i \cdot sp_i$
 2: calculate average service ratio $\bar{\mu} = \frac{\sum_{i=1}^C \mu_i}{C}$
 3: calculate service intensity $\rho_c = \frac{\lambda}{c\bar{\mu}}$
 4: **if** $\rho_c < 1$
 5: calculate the average forwarding delay \tilde{d} according to equation (10)
 6: **else**
 7: calculate \tilde{d} according to equation (12)
 8: **end if**
 9: calculate virtual bandwidth for direct link $C(X, Y_i, Y_i)$ according to equation (13)
 10: **return** $C(X, Y_i, Y_i)$

Algorithm 4 Calculate a record in $NVBT_X$ **Input:**

the node whose virtual bandwidth needs to be recalculated, Z_k

Output:

the virtual path's virtual bandwidth $C(X, Z_k)$
 1: set $n := 0$, $numerator := 0$, $denominator := 0$
 2: **for** every record $C(X, Y_j, Y_j)$ in $VPBT_X$ **do**
 3: **if** $C(Y_j, Z_k)$ exists in $NVBT_{Y_j}$ and $C(Y_j, Z_k) \neq 0$ **then**
 4: $denominator += C(X, Y_j, Y_j) / C(Y_j, Z_k)$
 5: $numerator += C(X, Y_j, Y_j)$
 6: $n ++$
 7: **end if**
 8: **end for**
 9: **if** $n \neq 0$ **then**
 10: set $C(X, Z_k) := numerator / (denominator + n)$
 11: **else** set $C(X, Z_k) := 0$
 12: **end if**
 13: **return** $C(X, Z_k)$

then it needs to recalculate virtual bandwidths in $VNBT$. We only refresh $C(X, Y_i, Y_i)$ when two nodes separate or meet. It should be pointed out that one node X does not need to recalculate all records of its $VNBT$ upon meeting or separating from another node. It only needs to update those $C(X, Z_j)$ records where $C(Y_i, Z_j)$ exists in $VNBT_{Y_i}$. Other records of $VNBT_X$ are irrelevant to node Y_i . The behavior of X when node X separates from node Y_j is formally described in Algorithm 2. Calculation of the virtual bandwidth for direct links and also the procedure for updating every record in $VNBT_X$ are described in Algorithms 3 and 4, respectively.

4.3 Message Forwarding Strategy in DTN-Balance

In DTNs, each node can set up connections with multiple nodes at the same time. Thus, we must decide to transfer which message to which node. To solve this multi-node multi-message problem, we create a transmitting priority database consisting of $tuples(message, message\ forwarding\ utility)$. The tuple consists of message m and a connected node Y 's *message forwarding utility (MFU)*. *MFU* is defined as follows:

$$MFU(m, Y) = \frac{C(Y, des_m)}{size(m)} \cdot \phi_m \quad (14)$$

In Eq. (14), des_m is the destination node of message m and $size(m)$ is message size of m , $C(Y, des_m)$ is the virtual bandwidth from Y to des_m , and ϕ_m is the meeting probability between current node and m 's destination node according to Eq. (17). To avoid long paths and lower the forwarding cost, we set max hop count for each message as 5. This is consistent with the architecture of social networks demonstrating Small World properties where the Six Degrees of Separation principle applies. In order to alleviate overdependence on few key nodes, we also apply a forwarding constraint. That is, the message forwarding is allowed only if the available buffer space of the next hop node is better than that of the current node. Otherwise, another relay node is chosen. The node sorts every possible tuple by its *MFU* and selects message that has the highest *MFU* and satisfies the hop number limit as well as free buffer size limit to transfer. We have designed our message forwarding strategy based on these principles which is formally described in Algorithm 5. After transferring m , if node Y is the destination, node X will delete message m with probability ϕ_m . The ϕ_m can be calculated according to the Eqs. (16) and (17).

When transferring message m to node Y , Y needs to decide whether or not to accept m . Node Y first determines it hasn't previously received the message m . Besides, in order to improve workload balance, node Y will refuse the forwarding request from those nodes whose buffer workload is lower.

When these conditions are met, it needs to make room for this message. If Y has enough buffer for this message, it accepts m . Otherwise, Y needs to decide whether or not to drop some messages to make room for m . Y makes this decision by sorting the messages in its buffer based on their dropping probabilities and then mark the messages whose dropping priority is bigger than the priority of m . The *Message Dropping Utility (MDU)* is defined as follows:

$$MDU(m) = \frac{size(m) \cdot D(m)}{m_r(T) \cdot \phi_m} \quad (15)$$

In Eq. (15), $size(m)$ is the size of message m , $m_r(T)$ is the remaining time to live for message m , $D(m)$ is the number of copies of message m that exist in the network, ϕ_m is the meeting probability between current node and m 's destination node which is calculated as follows:

$$\phi_m = \phi_m(old) + (1 - \phi_m(old)) \times P_{it} \quad (16)$$

where P_{it} is a constant and $P_{it} \in [0, 1]$.

The probability of meeting will decrease with the passage of time, if there is no encounter event. The Eq. (17) describes the aging process.

$$\phi_m = \phi_m \times \gamma^k \quad (17)$$

Algorithm 5 Message forwarding strategy**Input:**

all nodes connected, nc
 all messages in buffer, mc

Output:

```

void
1: create a empty tuple list  $tuples$ 
2: for current node  $X$  and every node  $Y$  in  $nc$  do
3:   for every message  $m$  in  $mc$  do
4:     if ( $MFU(m, X) < MFU(m, Y)$ ) && ( $freeBuffer(X) < freeBuffer(Y)$ ) && ( $m.max_{hop} > 1$ )
       && ( $Y$  is the biggest  $MFU$  value) then
5:       create tuple  $t(m, Y)$ 
6:       add  $t(m, Y)$  to  $TL$ 
7:     end if
8:   end for
9: end for
10: sort  $TL$  by  $MFU(m, Y)$  of tuple in descending order
11: while  $TL$  is not empty
12:   get and remove the first tuple of  $TL$ , as  $t(m, Y)$ 
13:   require  $Y$  to receive  $m$ 
14:    $Y$  decides whether accepts  $m$  using Algorithm 6
15:   if node  $Y$  refuse to accept  $m$  then
16:     continue
17:   end if
18:   transferring  $m$  to  $Y$ 
19: end while

```

where γ is a constant and $\gamma \in (0, 1)$, k is the number of time units that have elapsed from the last time of encounter event.

When all messages with higher dropping priority are marked, if the buffer is big enough for m , we accept m ; otherwise, we refuse m . Algorithm 6 describes the process of deciding whether to accept a message or not. The forwarding process of the DTN-Balance route is shown in Algorithm 7.

5 Performance Evaluation

In this section, in order to evaluate the effectiveness of the proposed DTN-Balance algorithm, we compare the performance of our proposed DTN-Balance schemes to existing representative routing schemes including Spray and Wait Router [8], ProphetRouterV2 (PRV2) [24], the SimBet [30] and the Bubble Rap [31].

We implement these routing protocols in the widely-adopted DTN simulator: ONE (opportunistic network environment simulator) [42]. The two realistic mobility data sets, UIM [40], Infocom2006 [41], are incorporated into ONE simulator. The characteristics of these datasets are described in Table 3. The time of simulation scenarios and the number of messages created depends on the input trace file. The ONE emulator puts discrete sequential contact events which are obtained from original trace files as inputs. The size of each message is set between 200k and 1M bytes and every 25s to 35s generates a message. The source and destination of each message are randomly chosen.

We measured the following metrics that are the generally accepted metrics of choice for the routing algorithms:

Algorithm 6 Check whether to accept a message or not**Input:**

size of the message to receive, $size_m$
 all messages in buffer, mc
 current node free buffer size, rbs
 forwarding node's free buffer size, fbs

Output:

void

```

1: if transferring or receiving other messages then
2:   refuse  $m$ 
3: else if already has  $m$  then
4:   refuse  $m$ 
5: else if  $fbs > rbs$  then
6:   refuse  $m$ 
7: end if
8: if  $rbs \geq size_m$  then
9:   accept  $m$ 
10: else create a empty marked message list  $mml$ 
11:   for every message  $mc_i$  in  $mc$  do
12:     if  $MDU_{mc_i} > MDU_m$  then
13:       add  $mc_i$  to  $mml$ 
14:        $mbs + = size_{mc_i}$ 
15:     end if
16:   end for
17:   if  $mbs + rbs \geq size_m$  then
18:     while  $size_m > rbs$  do
19:       find message  $mm_i$  with highest  $MDU$  in  $mml$ 
20:       drop  $mm_i$  in  $mc$ 
21:        $rbs + = size_{mm_i}$ 
22:     end while
23:   accept  $m$ 
24:   else
25:     refuse  $m$ 
26:   end if
27: end if

```

Table 3 Characteristics of mobility traces

Data set	UIM	Infocom2006
# Nodes	28	98
Duration (days)	20	3
Network type	Bluetooth	Bluetooth
Device	Google Phone	iMote
Granularity (s)	60	120

1. *Throughput* as the proportion of successfully delivered messages out of all generated messages;
2. *Cost* which is equal to the total number of forwarding occurrences;
3. *Efficiency* calculated as the ratio between throughput and number of forwards.

Algorithm 7 DTN-Balance message forwarding schedule when node X meet node Y

```

1: Initialization: obtain information from  $Y$  about  $Y$ 's  $NVBT_Y$ , update  $NVBT_X$ 
2: Direct delivery: deliver packets destined to  $Y$  in decreasing order of their  $MFU$  utility
3: Forwarding : for each message  $i$  in node  $X$ 's buffer do
4:   if  $i$  is already in  $Y$ 's buffer then
5:     ignore  $i$ 
6:   end if
7:   compute  $MFU(i, Y)$ 
8:   forward message according to  $MFU$  value and hop number limit and free buffer size limit
9:   end for
10: Receiving : for each receiving message  $j$  from node  $Y$ 's do
11:   if free buffer size of node  $X$   $rbs$  is lower than  $Y$ 's  $fb_s$ 
12:     continue
13:   end if
14:   compute  $MDU(j)$ 
15:   if  $X$  has free enough space then
16:     receive  $j$ ,  $Y$  deletes  $j$ 
17:   else if it can receiving  $j$  by dropping some messages then
18:     drop the message with bigger than  $j$ 's  $MDU$  until  $X$  has enough space for  $j$ , then receive  $j$  and  $Y$ 
       deletes  $j$ 
19:   else refuse  $j$ 
20:   end if
21: end for
22: Termination: end transfer and receive when out of radio rang or buffer has no message, update  $NVBT_X$ 

```

5.1 Evaluation of Efficiency in Single Message Copy Scenario

Single message copy means that although the receiving node is not the message destination, the forwarding node will delete the message from its buffer after transmission. Thus, there is only one copy of the message in the network.

5.1.1 Performance with Different Buffer Sizes

The performance of DTN-Balance is evaluated in infocom2006 when the size of buffer is from 5M and 1500M. In all cases, the value of TTL and link speeds are set as 23,040 s and 250 K/s, respectively. We also set the max hop count as 5 for each message during the whole experiment.

Delivery Ratio Figure 5a presents the throughput of the four routing protocols with the Infocom2006 trace as a function of the buffer size. We see that when the memory in each node increases, it results in a significant increase in the throughput. From delivery ratio point of view, we have the following ranking among various algorithms: ProphetV2 > DTN-Balance > SimBet > Bubble Rap when the buffer size is less than 100M, and ProphetV2 > SimBet > DTN-Balance > Bubble Rap when buffer size is more than 100M. DTN-Balance does not have the highest throughput because of its hop count limit and free buffer size limit constraints but the throughput degradation is minimal compared to the gain obtained in terms of the forwarding cost. Other routing protocols that have slightly better throughputs rely on nodes with a higher encounter probability. However, as the traffic rate goes up, a large number of packets pile up at the nodes, resulting in high queuing delays and packet drop ratios.

Cost Figure 5b presents the average delay for the four routing protocols using the Infocom2006 trace. We see that in terms of the cost we have the following apttern: Prophet >>

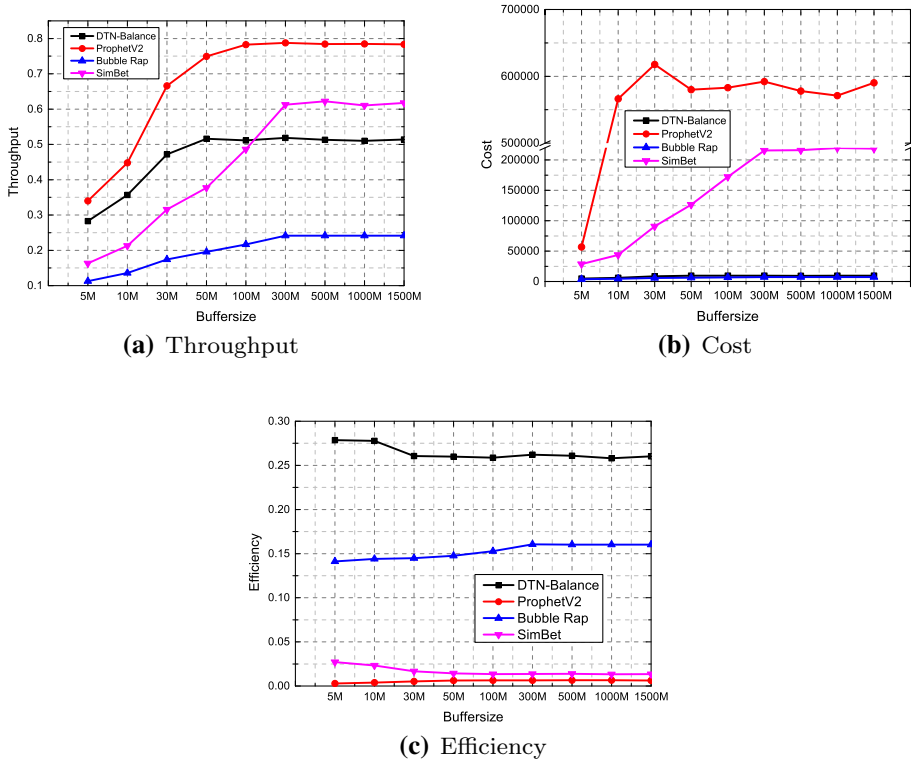


Fig. 5 Performance of DTN-Balance with different buffer sizes using the Infocom2006 trace

SimBet \geq DTN-Fast \approx Bubble Rap with Infocom2006. Although prophetV2 has the highest throughput (about is 1.6 time bigger than that of DTN-Balance), it also has the highest cost and its cost is 60 time bigger than that of DTN-Balance! Hence, ProphetV2 will rapidly deplete the energy of DTN nodes, especially the key active nodes which will result in network segmentation and decline of network survivability. As for Bubble Rap, it has the least forwarding cost since it assumes full knowledge of each nodes' connectivity and can find the global optimal solution. However, it also suffers from dependence on nodes with higher active degree which may be easily congested. Hence, it has the least throughput.

Efficiency Figure 5c plots the efficiency of the four protocols using the Infocom2006 trace. We find that in terms of the efficiency, the pattern is: ProphetV2 < SimBet < Bubble Rap < DTN-Balance with Infocom2006. DTN-Balance has highest efficiency since it does not fall in local optima trap and also limits invalid forwarding. What is more, compared to the second highest algorithm, Bubble Rap, the throughput of DTN-Balance shows overwhelming superiority. Compared to the ProphetV2, although the average throughput of DTN-Balance (0.466) is 5/7 of that of ProphetV2 (0.6807), the efficiency is almost 30 time higher. Thus the gain obtained in terms of efficiency is significant and the cost is acceptable.

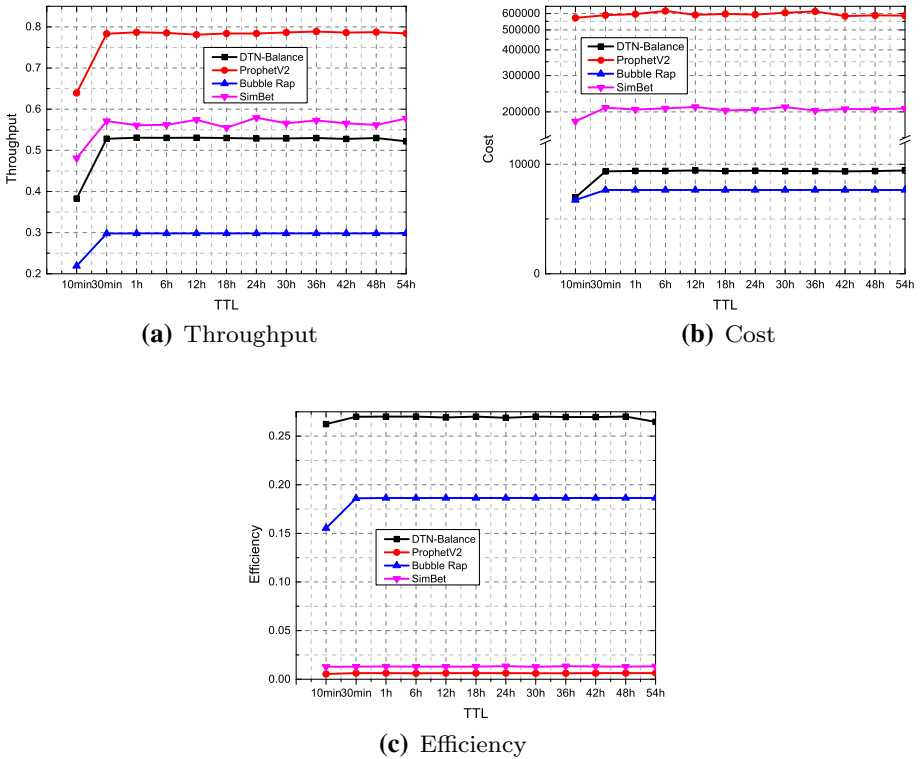


Fig. 6 Performance of DTN-Balance with different TTLs using the Infocom2006 trace

5.1.2 Performance with Different TTLs

Under different TTL values, the performance of four forwarding schemes are evaluated using Infocom2006. The connection speed and buffer size is set as 250 K/s and 200M, respectively.

Throughput Figure 6a presents the throughput of the four routing protocols using the Infocom2006 trace. We see that in terms of the throughput, we have ProphetV2 > SimBet > DTN-Balance > Bubble in Infocom2006. These results match those in Fig. 5a for similar reasons. The forwarding opportunities in the DTN are mainly determined by the encounter opportunities as well as the forward limit condition, which are independent from TTL. With the increase of TTL, the number of packets that can be successfully delivered does not increase accordingly.

Cost Figure 6b shows cost of the four routing protocols using the Infocom2006 trace. In general, we see that in terms of the cost, we have Prophet >> SimBet >> DTN-Balance > Bubble Rap with Infocom2006. These results match the results in Fig. 5b for similar reasons. One can observe that the cost of Bubble Rap always is the lowest with Infocom2006, DTN-Balance has a relatively higher cost than Bubble Rap but also has much higher throughput.

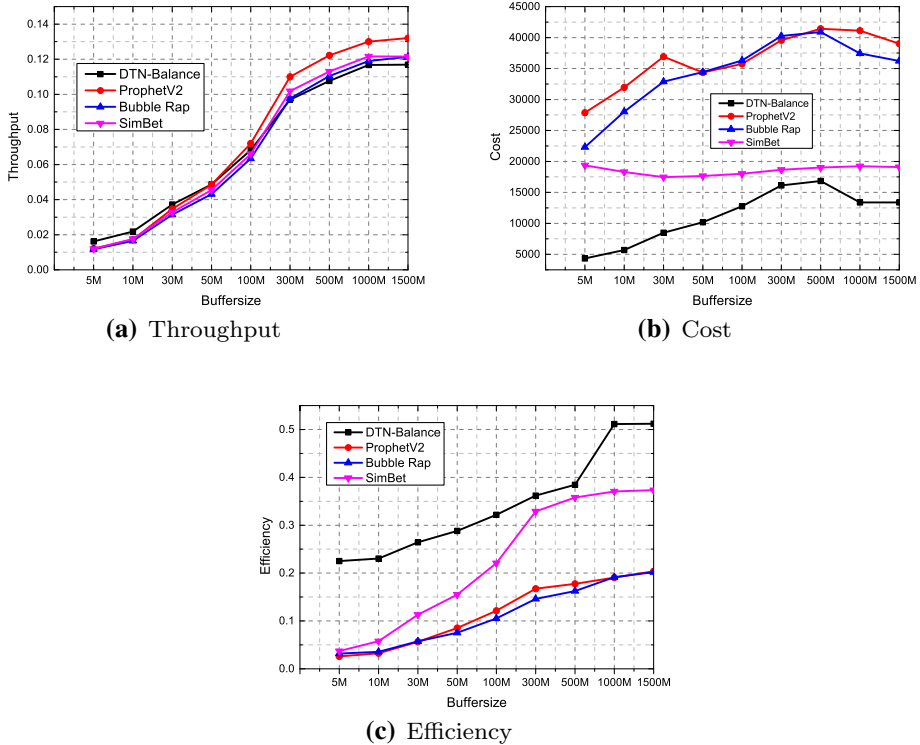


Fig. 7 Performance of DTN-Balance with different buffer sizes using the UIM trace

Efficiency Figure 6c shows the efficiency of the four routing protocols using the Infocom2006 trace. Again, this relationship is basically similar to that in Fig. 5c due to the similar reasons. Generally speaking, DTN-Balance has high throughput and low cost, thus its efficiency is best.

5.2 Evaluation of Efficiency in Multiple Message Copy Scenario

Multiple message copies means that if the receiving node is not the message destination, the forwarding node will not delete the message from its buffer. Thus, there might exist multiple copies of a message in the network.

5.2.1 Performance with Different Buffer Sizes

Under different buffer size values, the four forwarding schemes are evaluated using UIM trace. The connection speed and the TTL is set 250 K/s and 23,040 s, respectively.

Throughput Figure 7a presents the throughput of the four routing protocols using the UIM trace. We see that the average throughput follows the following pattern: ProphetV2 > DTN-Balance \approx SimBet > Bubble with UIM. Since UIM trace is a sparse dataset, the throughput is generally low, and there is little difference between different algorithms. The DTN-Balance has satisfactory throughput constrained only due to enforcing the forward limit condition.

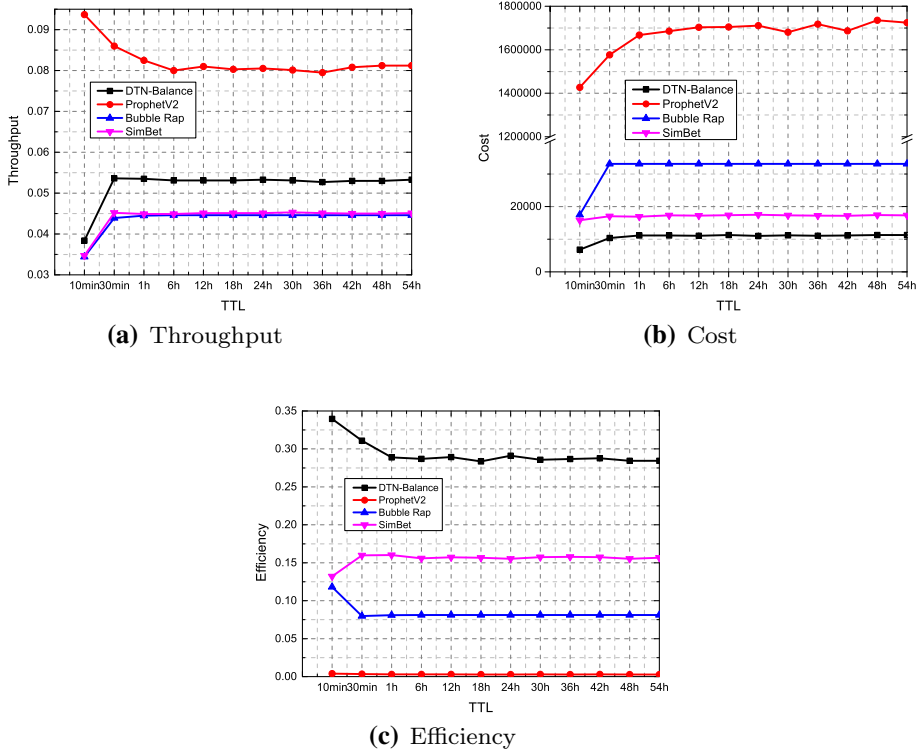


Fig. 8 Performance of DTN-Balance with different TTLs using the UIM trace

Cost Figure 7b shows the cost of the four routing protocols using the UIM trace. In general, we see the average cost follows the pattern: ProphetV2 > Bubble Rap > SimBet > DTN-Balance with UIM. The average cost of DTN-Balance is always the lowest with UIM. Its average cost is 11245, which is 39.3% better than that of SimBet (whose average cost is 18530). The average throughput of DTN-Balance is 0.07, which is 2.598% worse than that of SimBet (whose average throughput is 0.07189). The gain obtained in terms of cost is far greater than the slight loss of throughput.

Efficiency Figure 7c shows the efficiency of the four routing protocols using the UIM trace. We can see that the efficiency of DTN-balance is the best (efficiency value is 0.344) and is 53.84% better than the second best algorithm, SimBet (that is 0.2238). Generally speaking, DTN-Balance has the higher throughput, lower cost, and thus the best efficiency.

5.2.2 Performance with Different TTLs

Under different TTL values, the four forwarding schemes are evaluated using UIM trace. The connection speed and the buffer size is set 250 K/s and 50 M, respectively.

Throughput Figure 8a presents the through of the four routing protocols using the UIM. We see that the average throughput follows the pattern: ProphetV2 >> DTN-Balance > SimBet > Bubble. These results match those in Fig. 7a for the similar reasons.

Cost Figure 8b shows the average cost of the four routing protocols using the UIM. In general, we see that in terms of the average cost, we have DTN-Balance < SimBet < Bubble Rap < ProphetV2. These results match the results in Fig. 7b for similar reasons. We also find that the average cost of DTN-Balance always is the lowest with UIM. Due to sparse contact density in UIM dataset and lack of forwarding restrictions, other methods spend more cost to forward messages in the sparsely connected network.

Efficiency Figure 8c shows the efficiency of the four routing protocols using the UIM trace. The performance of the algorithms is similar to those in Figs. 5c and 7c due to similar reasons. Generally speaking, DTN-Balance fully utilizes all nodes' forwarding capacities and its forwarding schedule provisions result in superior performance. The average efficiency of DTN-Balance is 0.2932, which is 89% better than that of SimBet (0.1551) and demonstrates an overwhelming advantage over ProphetV2 and Bubble Rap.

Combining all the results obtained with various buffer sizes and TTLs, we can conclude that DTN-Balance has superior efficiency compared to the previous DTN routing algorithms. It achieves highest forwarding efficiency, and lowest forwarding cost as well as satisfactory throughput.

6 Conclusion and Future Work

In this paper, we proposed an efficient routing algorithm for message transfer among mobile nodes in a DTN. We developed the concept of *virtual bandwidth* to measure the forwarding capability from current node to destination node. Using the *virtual bandwidth* and *message properties*, we further proposed the message forwarding utility *MFU* and the message dropping utility *MDU* that are used to efficiently and intelligently forward packets on hop-by-hop basis until they reach their destinations. In order to improve the forwarding efficiency, we applied hop-count limits to avoid excessive long routes. Extensive analysis and trace-driven experiments using ONE simulator demonstrated the effectiveness of the proposed routing. As a future work, we plan to investigate how to apply network coding to further enhance the fairness and workload distribution in a DTN.

Acknowledgements This work is supported by the National Science Foundation of China under Grant No. 61572062, and the National Key Research and Development Program of China under Grant No. 2016YFB1000503.

References

1. Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of ACM SIGCOMM 03* (pp. 27–34).
2. Rodrigues, J. J. P. C., & Soares, V. N. G. (2015). 1-An introduction to delay and disruption-tolerant networks (DTNs). *Elsevier Journal of Advances in Delay-Tolerant Networks (DTNs)*, 2015, 1–21.
3. Jain, S., Fall, K. R., & Patra, R. K. (2004). Routing in a delay tolerant network. In *Proceedings of the SIGCOMM* (pp. 145–158).
4. Guo, S., & Keshav, S. (2007). Fair and efficient scheduling in data ferrying networks. In *Proceedings of the of ACM/CoNEXT*.
5. Balasubramanian, A., Levine, B. N., & Venkataramani, A. (2007). Dtn routing as a resource allocation problem. In *Proceedings of the of SIGCOMM*.
6. Lee, K., Yi, Y., Jeong, J., Won, H., Rhee, I., & Chong, S. (2010). Max-contribution: on optimal resource allocation in delay tolerant networks. In *Proceedings of the IEEE INFOCOM* (pp. 1–9).

7. Fan, X., Li, V. O. K., & Xu, K. (2014). Fairness analysis of routing in opportunistic mobile networks. *IEEE Transaction on Vehicular Technology*, 63(3), 1.
8. Pujol, J. M., Toledo, A. L., & Rodriguez, P. (2009). Fair routing in delay tolerant networks. In *IEEE INFOCOM 2009 proceedings* (pp. 837–845).
9. Qi, Y., Yang, L., Pan, C., et al. (2020). CGR-QV: A virtual topology DTN routing algorithm based on queue scheduling. *IEEE China Communications*, 17(7), 113–123.
10. Vazintari, A., & Cottis, P. G. (2016). Mobility management in energy constrained self-organizing delay tolerant networks: An autonomic scheme based on game theory. *IEEE Transactions on Mobile Computing*, 15(6), 1401–1411.
11. Pham, T. N. D., & Yeo, C. K. (2016). Detecting colluding blackhole and greyhole attacks in delay tolerant networks. *IEEE Transactions on Mobile Computing*, 15(5), 1116–1129.
12. Le, T., & Gerla, M. (2015). A load balanced social-tie routing strategy for DTNs based on queue length control. In *IEEE military communications conference* (pp. 383–387).
13. Vahdat, A., & Becker, D. (2000). *Epidemic routing for partially connected ad hoc networks*. Technical report CS-200006. Duke University.
14. Spyropoulos, T., Psounis, K., & Raghavendra, C. S. (2005). Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *WDTN 05: Proceeding of the 2005 ACM SIGCOMM workshop on delay-tolerant networking* (pp. 252–259).
15. Spyropoulos, T., Psounis, K., & Raghavendra, C. (2008). Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Transactions on Networking*, 16(1), 63–76.
16. Choochotkaew, S., Yamaguchi, H., & Higashino, T. (2018). BALANCE: A robust routing protocol in self-organized civilian DTN. In *2018 14th international conference on wireless and mobile computing, networking and communications (WiMob)*.
17. Sharif, H. Md. (2019). DTN routing protocols on two distinct geographical regions in an opportunistic network: An analysis. In *Springer wireless personal communications*.
18. Erramilli, V., Chaintreau, A., Crovella, M., & Diot, C. (2008). Delegation Forwarding. In *Proceedings of the ACM MobiHoc*.
19. Dubois-Ferriere, H., Grossglauser, M., & Vetterli, M. (2003). Age matters: Efficient route discovery in mobile ad hoc networks using encounter ages. In *Proceedings of the ACM MobiHoc* (pp. 257–266).
20. Nelson, S. C., Bakht, M., Kravets, R. (2009). Encounter based routing in DTNs. In *INFOCOM 2009*. New York: IEEE (pp. 846–854).
21. Liu, C., & Wu, J. (2009). An optimal probabilistic forwarding protocol in delay tolerant networks. In *MobiHoc09* (pp. 105–114).
22. Lindgren, A., Doria, A., & Scheln, O. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE*, 7(3), 19–20.
23. Burgess, J., Gallagher, B., Jensen, D., & Levine, B. N. (2006). MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proceedings of the IEEE INFOCOM* (pp. 1–11).
24. Grasic, S., Davies, E., Lindgren, A., & Doria, A. (2011). The evolution of a DTN routing protocol PRoPHETv2. In *CHANTS11* (pp. 27–30).
25. Link, J., Schmitz, D., & Wehrle, K. (2011). GeoDTN: Geographic routing in disruption tolerant networks. In *Proceedings of the IEEE GLOBECOM* (pp. 1–5).
26. Kurhinen, J., & Janatuinen, J. (2007). Geographical routing for delay tolerant encounter networks. In *Proceedings of the IEEE ISCC* (pp. 463–467).
27. Leguay, J., Friedman, T., & Conan, V. (2005). DTN routing in a mobility pattern space. In *Proceedings of the ACM SIGCOMM WDTN* (pp. 276–283).
28. Fan, J., Chen, J., Du, Y., Gao, W., Wu, J., & Sun, Y. (2013). Geo community based broadcasting for data dissemination in mobile social networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(4), 734–743.
29. Li, F., & Wu, J. (2009). MOPS: Providing content-based service in disruption-tolerant networks. In *Proceedings of the IEEE ICDCS* (pp. 526–533).
30. Daly, E. M., & Haahr, M. (2007). Social network analysis for routing in disconnected delay-tolerant MANETs. In *Proceedings of the ACM MobiHoc* (pp. 32–40).
31. Hui, P., Crowcroft, J., & Yoneki, E. (2008). Bubble rap: Social-based forwarding in delay tolerant networks. In *Proceedings of the ACM MobiHoc* (pp. 241–250).
32. Gao, W., Cao, G., La Porta, T., & Han, J. (2013). On exploiting transient social contact patterns for data forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 12(1), 151–165.
33. Wu, J., & Wang, Y. (2012). Social feature-based multi-path routing in delay tolerant networks. In *2012 proceedings IEEE INFOCOM* (pp. 1368–1376).
34. Chen, K., & Shen, H. (2015). DTN-FLOW: Inter-landmark data flow for high-throughput routing in DTNs. *IEEE/ACM Transactions on Networking*, 23(1), 212–226.

35. Tantai, J., Bai, Y., Zhao, Y., Liu, J., & Chen, W. (2015). Capacity routing based message priority for delay tolerant networks. In *Proceedings of the 24th international conference on computer communications and networks (ICCCN)*.
36. Haoran, S., Muqing, W., & Yanan, C. (2019). A community-based opportunistic routing protocol in delay tolerant networks. In *2018 IEEE 4th international conference on computer and communications (ICCC)*. New York: IEEE.
37. Qi, W., Song, Q., Wang, X., & Guo, L. (2017). Trajectory data mining-based routing in DTN-enabled vehicular ad hoc networks. *IEEE Access*, 5, 24128–24138.
38. Wang, W., Bai, Y., Feng, P., et al. (2018). DTN-Knca: A high throughput routing based on contact pattern detection in DTNs. In *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*, Tokyo (pp. 926–931). <https://doi.org/10.1109/COMPSAC.2018.00159>.
39. Wen, Z., et al. (2019). An adaptive probability prediction routing scheme in urban DTNs. In *2019 IEEE 25th international conference on parallel and distributed systems (ICPADS)*. New York: IEEE.
40. Nahrstedt, K., & Vu, L. (2012). *CRAWDAD data set uiuc/uim* (v. 2012-01-24). Downloaded from <http://crawdad.cs.dartmouth.edu/uiuc/uim>.
41. Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., & Chaintreau, A. (2009). *Data set cambridge/haggle/imote/infocom2006* (Online). <http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/infocom2006>.
42. The ONE simulator, an Available Website. <http://www.netlab.tkk.fi/tutkimus/dtn/theone/>.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Weitao Wang received B.S. degree in mathematics and applied mathematics from Hebei Normal University, Shijiazhuang, China, in 2010, and M.S. degree in applied mathematics from Fuzhou University, Fuzhou, China, in 2013. Currently he is a Ph.D. degree Candidate in School of Computer Science and Engineering at Beihang University. His research interests include DTN, Ad Hoc Networks and Cognitive Radio Networks.



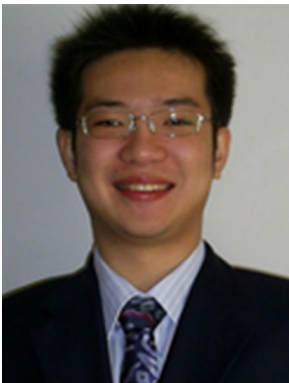
Yuebin Bai received his Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001. From 2001 to 2003, he was engaged in postdoctoral research at the College of Science and Technology at Nihon University, Tokyo, Japan. In 2003, he joined the faculty of Beihang University, Beijing, China, where he is currently a professor in School of Computer Science and Engineering. His research interests include Wireless Networks, Embedded and Real-Time Systems, System Virtualization and Cloud Computing.



Peng Feng received his B.S. degree in computer science from Xiangtan University, Xiangtan, China, in 2011. He received his MS degree in computer Science from Guangxi University, Nanning, China, in 2014. Currently he is a Ph.D. degree Candidate in School of Computer Science and Engineering at Beihang University. His research has focused on DTN, Ad Hoc Networks and Cognitive Radio Networks.



Jun Huang received the B.S. and M.S. degrees in School of Computer Science and Engineering from Beihang University, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science and engineering from Michigan State University, USA, in 2013. He is currently an assistant professor in the Center for Energy Efficient Computing and Applications (CECA), School of EEC, Peking University. He received the Best Paper Awards at the 18th IEEE International Conference on Network Protocols (ICNP) in 2010. His research interests include wireless networking and mobile systems.




Mo Sha is an Assistant Professor in the Department of Computer Science at Binghamton University—State University of New York, USA. He received his Ph.D. degree in Computer Science from Washington University in St. Louis, USA, in 2014. Prior to his Ph.D., he received a M.Phil. degree from City University of Hong Kong in 2009 and a B.Eng. degree from Beihang University in 2007. His research interests include Wireless Networks, Internet of Things, Embedded and Real-Time Systems, and Cyber-Physical Systems.



Jianpei Tantai received his B.S. degree in computer science from Hebei University of Technology, Tianjin, China, in 2013. He received his M.S. degree in School of Computer Science and Engineering from Beihang University, Beijing, China, in 2016. His research has focused on DTN, Ad Hoc Networks and Cognitive Radio Networks.

Affiliations

Weitao Wang¹ · Yuebin Bai¹  · Peng Feng¹ · Jun Huang² · Mo Sha³ · Jianpei Tantai¹

✉ Yuebin Bai
byb@buaa.edu.cn

Weitao Wang
weitaowang@buaa.edu.cn

Peng Feng
fengpeng@buaa.edu.cn

Jun Huang
jun.huang@pku.edu.cn

Mo Sha
mosha@binghamton.edu

Jianpei Tantai
JPTanTai@buaa.edu.cn

¹ School of Computer Science and Engineering, Beihang University, Beijing 100191, China

² Center for Energy Efficient Computing and Applications (CECA), Peking University, Beijing 100871, China

³ Department of Computer Science, State University of New York, Binghamton, NY 13902, USA