Pervasive and Mobile Computing 13 (2014) 150–163

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Utilizing correlated node mobility for efficient DTN routing

Eyuphan Bulut [a,b,*], Sahin Cem Geyik [a], Boleslaw K. Szymanski [a,c]

[a] *Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180, United States*
[b] *Cisco Systems, 2200 President George Bush Highway, Richardson, TX 75082, United States*
[c] *Społeczna Akademia Nauk ul Sienkiewicza 9, 90-113 Łódź, Poland*

ARTICLE INFO

Article history:
Received 28 November 2012
Received in revised form 14 September 2013
Accepted 20 September 2013
Available online 30 September 2013

Keywords:
Delay tolerant networks
Routing
Efficiency
Temporal correlation

ABSTRACT

In a delay tolerant network (DTN), nodes are connected intermittently and the future node connections are mostly not known. Therefore, effective forwarding based on limited knowledge of contact behavior of nodes is challenging. Most of the previous studies assumed that mobility of a node is independent from mobility of other nodes and looked at only the pairwise node relations to decide routing. In contrast, in this paper, we analyze the temporal correlation between the meetings of each node with other nodes and utilize this correlation for efficient routing. We introduce a new metric called *conditional intermeeting time (CIT)*, which computes the average intermeeting time between two nodes relative to a meeting with a third node. Then, we modify existing DTN routing protocols using the proposed metric to improve their performance. Extensive simulations based on real and synthetic DTN traces show that the modified algorithms perform better than the original ones.

1. Introduction

Delay tolerant networks (DTN) are wireless networks in which at any given time instance, the probability that there is an end-to-end path from a source to a destination is low due to the high mobility and low density of the nodes in the network. Routing of messages in such a challenging DTN environment is achieved opportunistically by utilizing the *store-carry-and-forward* paradigm at each node. Several DTN routing algorithms based on this paradigm have been proposed recently. In each, different techniques (single-copy [1–5], multi-copy [6–8], erasure coding [9,10]) with slightly different goals have been utilized. A survey of DTN routing algorithms can be found in [11].

Since DTNs consist of mobile agents that contact intermittently, recent studies on DTN routing have focused on the analysis of real mobility traces (human [12], vehicular [13] etc.) and utilized extracted characteristics of the mobile objects in the design of forwarding algorithms for DTNs.

Reviewing these designs and analyses, we have made the following observations. First, the future meetings of nodes can be predicted from their past relations using some distribution functions (e.g. log-normal [14,15]). Second, most of the previous routing protocols assume that meetings of a node with other nodes are independent from each other. Some algorithms (e.g. Bubblerap [16]) implicitly consider the dependency between the node meetings thanks to their designs which use real traces, however, there is no analysis and explicit usage of temporal correlation between the meetings of two nodes with a third node. Third, the mobility of many real objects are non-deterministic but periodic [17]. For example, in

* Corresponding author at: Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180, United States. Tel.: +1 4694225653.
E-mail addresses: ebulut@cisco.com, bulute@cs.rpi.edu (E. Bulut), geyiks@cs.rpi.edu (S.C. Geyik), szymansk@cs.rpi.edu (B.K. Szymanski).

1574-1192/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.pmcj.2013.09.007

a cyclic MobiSpace, if two nodes were frequently in contact at a particular time in previous cycles, then they are likely to meet around the same time in the next cycle.

The above observations motivated us to study temporal correlation between the node meetings for designing more efficient routing algorithms. Hence, we introduce a new link metric, *conditional intermeeting time (CIT)*, that computes the average intermeeting time between two nodes relative to a meeting with a third node using only local knowledge of the past contacts. Note that this definition makes more sense in the context of routing because it refers to message holding time on a given node during message routing.

We analyze CIT and discuss when and why it is beneficial in providing accurate information to nodes making routing decisions. Different than our initial work [18], we also present statistical results from four different data sets (RollerNet [15], Cambridge [19], Haggle [20], MIT [21]) which contain the contact traces of real objects logged during real life experiments.

We then propose modifications to the existing DTN routing protocols using the proposed metric and demonstrate how their performance improves as the result. First, for the algorithms which route messages over shortest paths (SP) [22,23], we propose to use CIT rather than standard intermeeting times (SIT) and route the messages over conditional shortest paths (CSP). Second, for the algorithms which make message forwarding decisions depending on a delivery metric (we call them metric-based forwarding algorithms), we propose to use CIT as a secondary delivery metric and allow the forwarding of messages if and only if both the algorithm's original delivery metric and CIT agree to forward the message to the encountered node. Through simulations, we show the benefit of the proposed approach. In this extended version, we added new simulation results and comparisons over our initial work [18]. In addition to real DTN traces, we also utilized synthetic and large-scale traces for simulations. Moreover, we added new results showing the superiority of the proposed approach over other popular algorithms (i.e., SimBet and CREST) and added new graphs showing the effects of some important parameters (buffer space at nodes, message generation interval, and total node count) on the results.

The remainder of the paper is organized as follows. In Section 2, we describe the proposed metric (CIT) in detail and in Section 3, we provide its analysis. In Section 4, we describe how it can be used to modify existing DTN routing algorithms for performance improvement. In Section 5, we present simulation results. Finally, we offer conclusions and outline the future work in Section 6.

2. Conditional intermeeting time

Recently, the research community working on routing algorithms in DTNs has focused on the analysis of real mobility traces to understand the main characteristics of mobile objects. Several experiments in different environments (office [14], conference [20], city [19], skating tour [15]) with different objects (human [12], bus [13]) and with variable number of attendants were performed. From the analysis of the data sets collected during these experiments, significant results about the aggregate and pairwise mobility characteristics of real objects were obtained and different kinds of algorithms using different metrics developed for efficient routing of messages in DTNs. For example, in SimBet [24], a joint utility metric based on social similarity and egocentric betweenness of nodes is used. In BubbleRap [16] two rankings (global and local) is used to compute each node's popularity (i.e., connectivity) in the local community and the entire society. Routing of messages are then performed via nodes with high utility values. Moreover, in LocalCom [25], a community-based epidemic forwarding scheme, which first detects the community structure of the network and forwards the messages to each community through gateways, is proposed.

Even though the previous studies modeled the node relations (i.e., intermeeting times) using different distributions (exponential [7,26], log-normal [15]) and developed their routing metrics accordingly, they assumed that the meetings of a node with other nodes is independent of each other. The future meetings of two nodes are predicted looking at the meetings of only these two nodes in the past. In [14], one additional attribute (the time passed since the last meeting) is also taken into account for more accurate prediction. However, as we show with statistics from real DTN traces, the meetings of a node with other nodes may not be independent from each other (i.e., meetings are correlated), thus, prediction of future node relations can be further improved with the analysis of temporal correlation between node meetings. In [5], each node establishes a community of nodes with whom it frequently meets to use for routing decisions when this node meets a message carrier. In contrast, in this paper, we consider a sequence of meetings of two specific nodes and use the statistics about subsequent meetings of those nodes with the destination to make routing decisions.

We introduce a new metric called *conditional intermeeting time (CIT)*, to define the node relations more precisely within the context of routing. This metric computes the average intermeeting time between two nodes relative to a meeting with an intermediate node using only the local knowledge of the past contacts.

The proposed metric can provide higher accuracy of prediction of delivery time, especially if the nodes move in a cyclic MobiSpace [17]. According to the definition of a cyclic MobiSpace, if two nodes contacted frequently at a particular time in previous cycles, the probability that they will be in contact around the same time in the next cycle is high. In Fig. 1, a sample cyclic MobiSpace with three objects is illustrated. The fully repeating motion cycle is 12 time units. In this example, the discrete probabilistic contacts between A and M happen every 12 time units (1, 13, 25 ...) and the discrete probabilistic contacts between A and B occur every 6 time units (2, 8, 14 ...). When we consider the intermeeting time between nodes A and B, we can expect that node A will forward its message to node B in 3 time units (since a message can arrive at A at any time within 6 s), however CIT of A with B based on the condition that it has met (received the message from) node M let us know that the message will be forwarded to node B within 1 time unit.

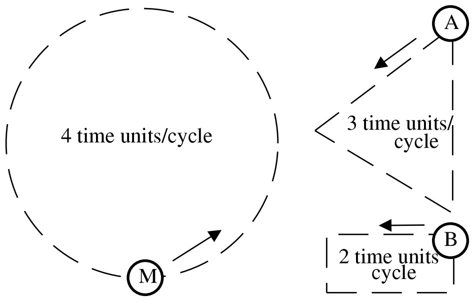


Fig. 1. An example cyclic MobiSpace with a common motion cycle of 12 time units.

Algorithm 1 updateSIT (node m, time t)

```
1: if $m$ is seen first time then
2:     firstTimeAt[$m$] $\leftarrow t$
3: else
4:     increment $\beta_m$ by 1
5:     lastTimeAt[$m$] $\leftarrow t$
6: end if
7: for each neighbor $i \in N$ do
8:     $\tau_s(i) \leftarrow$ (lastTimeAt[$i$] – firstTimeAt[$i$] ) / $\beta_i$
9: end for
```

Each node in a DTN can compute its SIT and CIT with other nodes using its contact history. In Algorithm 1, we show how a node, s, can compute these metrics from previous node meetings. The notations we use in this algorithm (and also throughout the paper) are listed below with their meanings:

- $\tau_A(B|M)$: Average time it takes for node A to meet node B after meeting with node M. If $B = M$, the notation (in short $\tau_A(B)$) shows the standard intermeeting time (SIT) between nodes A and B.
- S: $N \times N$ matrix where $S\ (i, j)$ is the sum of all samples of conditional intermeeting times with node j relative to the meeting with node i. Here, N is the neighbor count of current node (i.e., $N(s)$ for node s).
- C: $N \times N$ matrix where $C\ (i, j)$ is the number of all conditional intermeeting time samples with node j relative to meeting with node i.
- β_i: Total number of meetings with node i.

Algorithm 2 updateCIT (node m, time t)

```
1: for each neighbor $j \in N$ and $j \neq m$ do
2:     start a timer $t_{mj}$
3: end for
4: for each neighbor $j \in N$ and $j \neq m$ do
5:     for each timer $t_{jm}$ running do
6:         $S[j][m]$ += time on $t_{jm}$
7:         increment $C[j][m]$ by 1
8:     end for
9:     if $S[j][m] \neq 0$ then
10:        $\tau_s(m|j) \leftarrow S[j][m]\ /\ C[j][m]$
11:    end if
12:    delete all timers $t_{jm}$
13: end for
```

To find the CIT $\tau_A(B|M)$, each time node A meets node M, it starts a different timer for B. When it meets node B, it sums up the values of these timers before deleting them. The value of $\tau_A(B|M)$ is then calculated by dividing the current total of collected timer values by the number of timers used so far. In Algorithm 2, we explain this procedure. Note that when the node meets any node m, a timer is started for every other possible node (lines 1–3). Since meeting with node m will also end the conditional meeting time process for some other nodes, the corresponding timers whose clock has started at the meeting with other nodes but is scheduled to end at the meeting with m stop and their values are recorded (lines 4–8). CIT values are then updated for all possible cases and the timers are deleted (lines 9–12). We can also use a sliding window with an appropriate size over the past contacts [23] and take into account the edge effects [13] to make the computation more local and current. Moreover, we do not consider the contact durations in these computations since inter-contact times are

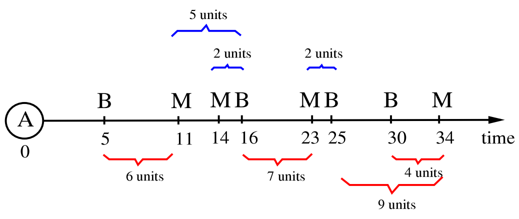


Fig. 2. Example meeting times of node A with nodes B and M. Upper and lower values are used to compute $\tau_A(B|M)$ and $\tau_A(M|B)$, respectively.

usually much longer than contact times in real DTNs. However, if the last assumption does not hold, it is easy to modify all computations accordingly.

Consider the sample meeting times of a node A with its neighbors B and M in Fig. 2. Node A meets with node B at times $\{5, 16, 25, 30\}$ and with node M at times $\{11, 14, 23, 34\}$. Following the procedure described above, we find that $\tau_A(B|M) = (5 + 2 + 2)/3 = 3$ time units and $\tau_A(M|B) = (6 + 7 + 4 + 9)/4 = 6.5$ time units while $\tau_A(B) = 8.33$ time units and $\tau_A(M) = 7.67$ time units.

3. Analysis

In this section, we give the analysis of CIT and show why it is significant in accurate prediction of future meetings within the context of routing.

Assume that node A has two different contacts, B and M, and assume that the intermeeting time of node A with M is well represented by a random variable X_{AM} with the CDF $D_{AM}(x)$ and pdf $d_{AM}(x) = D'_{AM}(x)$.

Then, let $X_{AB|M}$ denote the random variable representing the CIT of A with B under the condition that it has met M before meeting B, with CDF $D_{AB|M}(x)$ and pdf $d_{AB|M}(x)$. Let also $X^t_{(AM)B}$ denote the random variable of time between the current meeting of A with M and the next meeting of A with B, with CDF $D^t_{(AM)B}(x)$ and pdf $d^t_{(AM)B}(x)$ given that t time units has passed since the last meeting of A with B and the current meeting of A and M. Then:

$$\tau_A(B|M) = \int_{t=0}^{\infty} d^t_{(AM)B} E[X^t_{(AM)B}]dt.$$

In general, we can write in the case of such a meeting of A and M:

$$X_{AB|M} = t + X^t_{(AM)B}.$$

Here, if A–M meeting is independent of A–B meeting, meaning that $D_{AB|M} = D_{AB}$, then:

$$D^t_{(AM)B}(x) = \frac{D_{AB}(x+t) - D_{AB}(t)}{1 - D_{AB}(t)}$$

$$d^t_{(AM)B}(x) = \frac{d_{AB}(x+t)}{1 - D_{AB}(t)}.$$

Hence:

$$E[X^t{}_{(AM)B}] = \frac{\int_{x=0}^{\infty} x d_{AB}(t+x)dx}{1 - D_{AB}(t)}.$$

Then, using $[z(1 - D(z))]' = 1 - D(z) - zD'(z)$, we get:

$$E[X^t_{(AM)B}] = \frac{\int_{z=t}^{\infty}(1 - D_{AB}(z))dz}{1 - D_{AB}(t)}.$$

There have been several distribution functions studied in previous work to model the pairwise intermeeting times. Using a distribution fitting software (EasyFit [27]), we also checked the fitness of several distribution functions including exponential, Pareto and log-normal distribution and observed that intermeeting time (i.e., X_{AB}) between nodes fits well with log-normal distribution. The analysis here can be updated for other distribution functions and is orthogonal to the distribution function assumed. In case of log-normal distribution, we get:

$$E[X^t_{(AM)B}] = e^{\mu + \frac{\sigma^2}{2}} \frac{1 - \text{erf}\left(\frac{\ln t - (\mu + \sigma^2)}{\sigma\sqrt{2}}\right)}{1 - \text{erf}\left(\frac{\ln t - \mu}{\sigma\sqrt{2}}\right)} - t$$

where *erf* is the error function and μ and σ are mean and variance of X_{AB}, respectively.

Thus, if meeting of A with M is not correlated with meeting of A with B, $E[X^t_{(AM)B}]$ depends on t. However, considering the behavior of people in real life, temporal correlations often arise between A's meetings with M and B (so A–B meeting

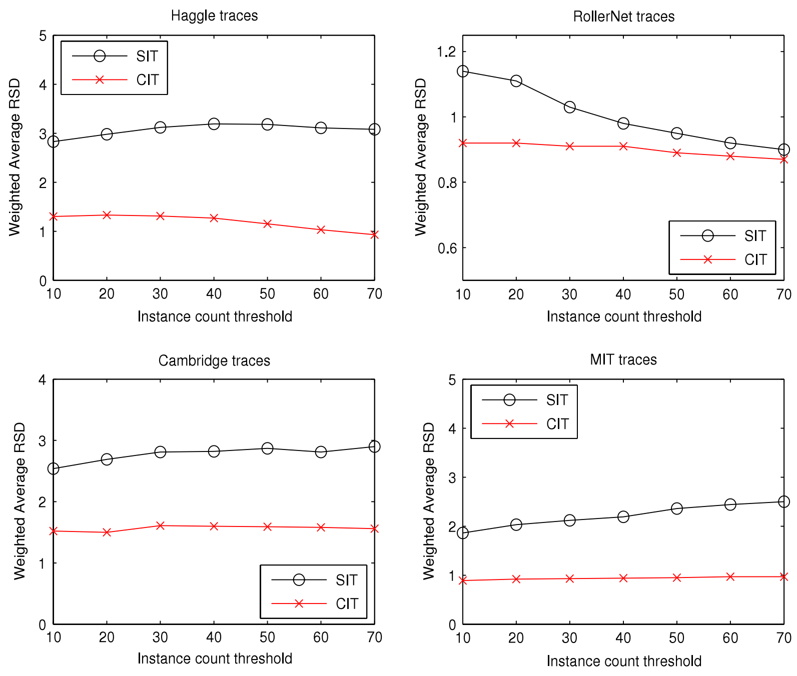


Fig. 3. Weighted average of relative standard deviation (RSD) for SIT and CIT in different datasets.

after A met M depends on A–M meeting time), yielding a different $d^t_{(AM)B}$ than it is in the uncorrelated case. This is the case, for example, when after A meets M, another stage of A's journey starts and the length of this stage is largely independent of what happened earlier. Consider meetings of a man who every morning goes from home to work. After meeting his family members (while leaving home), he meets later his office peers. Yet, on the way to his office, he meets the security guard at the gate of his workplace a few moments before meeting his office peers. In other words, the meetings of this man with his office peers are defined by the time when he met the guard but independent of how long it took him to meet the guard after leaving home. E.g, if the trip from the gate to the office is normally distributed with the average and variance of 1 time unit, then $X^t_{(AM)B} \approx N(1, 1)$ is independent of t, the travel time of the man from home to the gate. Therefore, we compute and use the average of the time passed from A–M meeting to A–B meeting based on currently collected samples from encounter history.

Next, we present some statistics from available real DTN traces to show (i) the advantage of CIT over SIT and (ii) temporal correlation between the meetings of nodes.

For the first one, we found the answer to the question 'What would the average error in predicting the future meetings be if the nodes could know their $\tau_A(B)$ and $\tau_A(B|M)$ values in advance?'. In SIT, for a given (A, B), this refers to standard deviation (std) of SIT instances from their mean which is $\tau_A(B)$. Similarly, in CIT, for a given (A, M, B) tuple, this refers to standard deviation (std) of CIT instances from their mean which is $\tau_A(B|M)$. However, to compute the average of this error for all possible and different (A, B) (in SIT) and (A, M, B) (in CIT) tuples, we computed the 'relative standard deviation (RSD)', computed as std/mean, and took the weighted average (WA) of these RSD values. More formally, WA-RSD for SIT is computed as follows:

$$\text{WA-RSD(SIT)} = \left(\frac{\sum\limits_{\forall A \neq B} \left[\frac{std(A,B)}{\tau_A(B)} |\tau_A(B)| \right]}{\sum\limits_{\forall A \neq B} |\tau_A(B)|} \right)$$

where $|\tau_A(B)|$ denotes the instance counts used in computing the $\tau_A(B)$ and $std(A, B)$ denotes the standard deviation of these instances.

Similarly, for CIT, WA-RSD is computed as:

$$\text{WA-RSD(CIT)} = \left(\frac{\sum\limits_{\forall A \neq M \neq B} \left[\frac{std(A,M,B)}{\tau_A(B|M)} |\tau_A(B|M)| \right]}{\sum\limits_{\forall A \neq M \neq B} |\tau_A(B|M)|} \right).$$

To make these results statistically reliable, we only considered corresponding tuples with instance counts higher than a threshold and looked at the change in their value for different thresholds as well. Fig. 3 shows these results for different

Table 1
Ratio of all (A, B) pairs whose CIT values ($\tau_A(B|M_i)$) for different M_is pass the ANOVA test.

Instance count threshold	Haggle	RollerNet	Cambridge	MIT
10	96%	51%	64%	56%
20	95%	42%	58%	54%

thresholds in four different datasets. Clearly, WA-RSD values of the *CIT* metric are smaller than WA-RSD values of the *SIT* metric in each dataset. Only for RollerNet traces [15], the results get closer for some thresholds. Consequently, these results show that the *CIT* metric can provide more accurate prediction than the *SIT* metric for different environments.

To measure the temporal correlation between the meetings of nodes, we compare $\tau_A(B|M)$ values for different M's. As the above analysis shows, if A's meetings with M are random, then $E[X^t_{(AM)B}]$ so the $\tau_A(B|M)$ should be the same for different M's. To check if this is the case, we applied the ANOVA test on the CIT values of different M values. For each (A, B) pair, we found $\tau_A(B|M_0), \tau_A(B|M_1), \dots \tau_A(B|M_k)$ values (and also all the instance values used to compute each mean $\tau_A(B|M_i)$) for all applicable M_i values ($0 \le i \le k$). Then, we applied the ANOVA test to learn whether these $\tau_A(B|M_i)$ values and also their instance value distribution differ from each other significantly (with $\alpha = 0.05$) for given pair (A, B). Table 1 shows the ratio of all (A, B) pairs which pass this ANOVA test in different datasets. Clearly, the results indicate that for a remarkable amount of (A, B) pairs, CIT values computed using different M values are significantly different from each other. This indicates that the identity of met intermediate node (M) is significant (in predicting the A's future meeting with B) for all datasets. Thus, there is a temporal correlation between the meetings of a node with other nodes. If there were no such correlation, $\tau_A(B|M)$ would have been the same (or close) for different M's for given pair (A, B), causing the failure in ANOVA tests. In contrast, we observe that these values are significantly different from each other.[1]

4. Proposed algorithms

In this section, we present two different applications of CIT to the existing DTN routing algorithms. First, we look into the shortest path based routing algorithms and propose to use conditional shortest paths to route messages. Second, we propose to revise message forwarding decisions of metric-based forwarding algorithms by including CIT.

4.1. Shortest path based routing

4.1.1. Overview

Shortest path routing (SPR) protocols for DTNs are based on the designs of routing protocols for traditional networks. The links between each pair of nodes are assigned a cost and messages are forwarded over the shortest paths between the source and the destination. Furthermore, the dynamic nature of DTNs is also addressed in these designs. Two of the common metrics used to define the link cost are minimum expected delay (MED [22]) and minimum estimated expected delay (MEED [23]). These metrics compute the expected waiting time plus the transmission delay between each pair of nodes. However, while the former uses the future contact schedule, the latter uses only observed contact history.

In SPR, a routing decision can be made: (i) at source node, (ii) at each hop, and (iii) at each contact with other nodes. The utilization of recent contact information increases from the first to the last one improving the quality of the forwarding decisions; however, more processing resources are used as the routing decisions are made more frequently.

The suitability of SPR algorithms for DTNs and the scalability and complexity of their designs have been already discussed in [22,23], hence, in this paper, we focus on the enhancements of the performance of SPR algorithms achieved by utilizing our metric (CIT), rather than using SIT. To this end, in the rest of this section, we show the necessary changes to the current designs of SPR algorithms.

4.1.2. Network model

We model a DTN as a graph $G = (V', E')$ where the mobile nodes are represented by vertices (V') and the possible connections between these nodes are represented by the edges. Unlike previous DTN graph models, since CIT considers node relations with respect to a third node, we define V' and E' sets in a different way. Given V is the set of all node names and $N(i)$ denotes the set of other nodes that meet with node i (i.e. neighbors of node i):

$$V \subseteq V \times V \quad \text{and} \quad E' \subseteq V' \times V' \quad \text{where,}$$
$$V' = \{(i_j) \mid \forall j \in N(i)\}$$
$$E' = \{(i_j, k_l) \mid i = l\}$$
$$\text{where, } w'(i_j, k_l) = \begin{cases} \tau_i(k|j) & \text{if } j \neq k \\ \tau_i(k) & \text{otherwise.} \end{cases}$$

[1] It might also be interesting to analyze the divergence of ANOVA test results in different datasets and its impact on simulation results, which we will study in our future work.

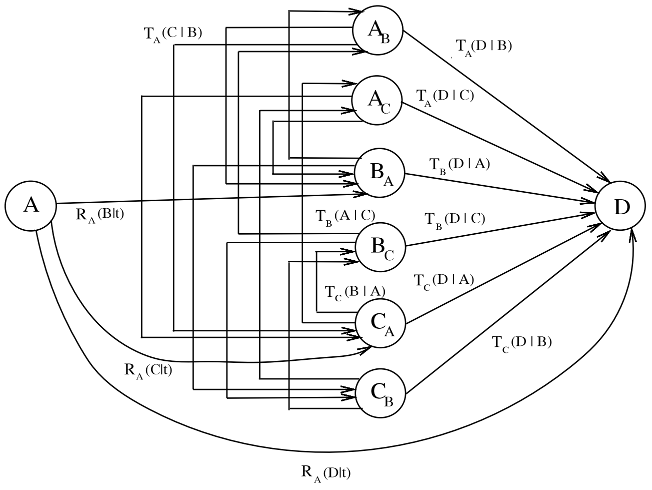


Fig. 4. A sample DTN graph with 4 nodes where A is the source and D is the destination.

4.1.3. Conditional shortest path routing

Our algorithm basically finds conditional shortest paths (CSP) for each source–destination pair and routes the messages over these paths. We define the CSP $\langle n_0, n_1, \ldots, n_{d-1}, n_d\rangle$ as follows:

$$CSP(n_0, n_d) = \min \left\{ \Re_{n_0}(n_1|t) + \sum_{i=1}^{d-1} \tau_{n_i}(n_{i+1}|n_{i-1}) \right\}.$$

Here, t represents the time that has passed since the last meeting of n_0 with n_1 and $\Re_{n_0}(n_1|t)$ is the expected residual time to the next meeting of n_0 and n_1 given that they have not met in the last t time units. $\Re_{n_0}(n_1|t)$ can be computed as in [14] with parameters of distribution representing the intermeeting time between n_0 and n_1. It can also be approximated iteratively from the observed intermeeting times of n_0 and n_1. Assume that n_0 observed k intermeeting times with n_1 in the past. Let $\tau^1_{n_0}(n_1), \tau^2_{n_0}(n_1), \ldots \tau^k_{n_0}(n_1)$ denote these values. Then, at time t, the iterative computation of $\Re_{n_0}(n_1|t)$ can be defined formally as follows:

$$\Re_{n_0}(n_1|t) = \frac{\sum_{s=1}^{k} f^s_{n_0}(n_1)}{|\{\tau^s_{n_0}(n_1) \geq t\}|} \quad \text{where,}$$

$$f^s_{n_0}(n_1) = \begin{cases} \tau^s_{n_0}(n_1) - t & \text{if } \tau^s_{n_0}(n_1) \geq t \\ 0 & \text{otherwise.} \end{cases}$$

If none of the k observed intermeeting times is bigger than t (this case is less likely to occur as the contact history grows), $\Re_{n_0}(n_1|t)$ is set to 0, which is a good approximation.

Each node forms the DTN using the aforementioned network model and collects SIT and CIT information of other nodes via epidemic link state protocol as it is described in the original study [23]. Note that, thanks to the design of the aforementioned network model which provides only valid CSP paths between nodes, running the Dijkstra or Bellman–Ford algorithm on the current graph structure gives us the correct CSPs for each source–destination pair.

In Fig. 4, we show a sample DTN graph where all mobile nodes A to D meet with each other and we set the source node to A and destination node to D (unused edges are not shown for brevity). Note that the graph includes all possible paths from A to D and does not contain unlikely edges like (C_D, D_A). Hence, only the correct τ values will be added to the path calculation. To solve the CSP problem however, we add one vertex for source S and one vertex for destination node D. We also add outgoing edges from S to each vertex $(i_S) \in V'$ with weight $\Re_S(i|t)$. Furthermore, for the destination node, D, we only add incoming edges from each vertex $i_j \in V'$ with weight $\tau_i(D|j)$ and from S with weight $\Re_S(D|t)$.

Running Dijkstra's shortest path algorithm on G' given the source node S and destination node D will give the shortest conditional path. In G, $|V'| = O(|V|^2)$ and $|E'| = O(|V^3|)$, thus, Dijkstra's algorithm will run in $O(|V|^3)$ (with Fibonacci heaps) while computing the original shortest paths (with SIT and simple DTN graphs) takes $O(|V|^2)$.

Using CIT instead of SIT only requires (over the original design) extra space to store the CIT values and additional processing, as complexity of running Dijkstra's algorithm increases from $O(|V|^2)$ to $O(|V|^3)$. We believe that in current DTNs, wireless devices have enough storage and processing power not to be unduly taxed with such an increase. Moreover, to lessen the burden of collecting and storing link weights, an asynchronous and distributed version [28] of the Bellman–Ford algorithm can be used.

4.1.4. Why CSPR offers better performance?

The difference between CSPR and SPR path definitions is that CSPR defines the link weights based on the previous node while SPR does not. In SPR, the path (SP) can be defined as:

$$SP(n_0, n_d) = \min \left\{ \Re_{n_0}(n_1|t) + \sum_{i=1}^{d-1} \frac{\tau_{n_i}(n_{i+1})}{2} \right\}.$$

After the first hop, since the message can reach the nodes at any time between their interaction with other nodes, each link weight can be approximated as $\frac{\tau_{n_i}(n_{i+1})}{2}$. However, the link weight can better be defined via $\tau_{n_i}(n_{i+1}|n_{i-1})$. If there is no temporal correlation between, say, n_1's meetings with n_0 and n_2, then $\tau_{n_1}(n_2|n_0)$ converges to $\frac{\tau_{n_1}(n_2)}{2}$, but if such correction exists (which is often the case as shown in statistics from real DTN traces) then these values will be different and routing via CSPR will therefore be faster.[2]

4.2. Metric-based forwarding algorithms

4.2.1. Overview

A common method of routing in DTNs is to forward the message to the encountered node that is more likely to meet with destination than the current message carrier. However, making effective forwarding decisions in single-copy based routing in DTNs is a challenging task. When two nodes meet, one of them forwards a message to the other one if it decides that the message will have a higher chance to be delivered to the destination at the other node.

In previous work, depending on the observed contact history between nodes, several metrics have been used to define the delivery quality of nodes. Popular ones include encounter frequency [2], time elapsed since last encounter [29,30], residual time [14] and social similarity [24,16].

4.2.2. Proposed modification

In most of the previous work, meetings of a node with other nodes are assumed independent from each other and the forwarding decision at the encounter of two nodes is made depending on their individual relations with the destination node. In some algorithms such as [2,30], with additional processing (i.e. applying transitivity) on pairwise meetings, more accurate metrics are used to reflect the effect of other nodes on the delivery quality of a node. However, these improvements can also be applied to all other metrics, including the one introduced in this paper. Our contribution is the introduction of a new metric having this property by default in its basic definition.

To make forwarding decisions of these algorithms more effective, thus to improve their performance, we propose to use CIT as an additional delivery metric. That is, when two nodes meet, they will also compare their CIT with destination (depending on the condition that they met each other). If the current carrier of the message learns that the other node also has a shorter remaining time (according to CIT) to meet the destination than itself, the message is forwarded. This additional condition eliminates forwardings that based on CIT became harmful and if executed would decrease the delivery probability. Simulation results confirm this conclusion, as the delivery rates are preserved and simply unbeneficial forwardings are not performed. Therefore, more effective forwarding decisions are made so that the cost of message delivery declines while the delivery ratio and average delay are maintained (in some cases, even the delivery ratio increases and average delay decreases).

4.2.3. Why modification offers better performance?

With the addition of CIT as the second forwarding condition, forwarding decisions are made depending on both the pairwise node relations between A and B (due to the metric of the original algorithm) and also possible temporal correlations between the meetings of A with nodes M and B. If there is no such correlation, CIT often supports the forwarding if the original metric also supports the forwarding. However, when this correlation is strong, CIT offers more accurate prediction and it may indicate that at the time of the meeting the forwarding is no longer beneficial. Thus, the statistically harmful forwarding decisions without considering possible correlations are prevented. Even though addition of CIT makes modified algorithms more selective, the messages are forwarded to or stay with the nodes which have higher delivery probability at the time of the meeting with node M.[3] Thus, delivery performance stays similar or improves while the cost (i.e., the number of forwardings) decreases, yielding better routing efficiency.

[2] It is should be noted that the values of the τ function are approximated iteratively. However they are used to select the minimum delay paths, so the error of selection is bounded by the error of approximation. In other words, if iterative averages are close to each other, so are the real averages, thus wrong selection will have a small impact on performance. Moreover, such an error of selection arises in all routing methods using the iteratively approximated averages. Thus, this error does not negate the improvements of CSPR.

[3] Note that the path to delivery in modified algorithms can be totally different than the path in original algorithms. In the original algorithm A may forward the message to M_1 but in the modified version, A may skip M_1 due to the unsatisfied CIT condition and later forward the message to M_2 which satisfies both conditions. The remaining paths of the message towards the destination in both cases are likely to continue to be partially disjoint.

5. Performance evaluation

To evaluate the performance of proposed algorithms, we have built a Java based custom DTN simulator. It uses either the traces of real objects from real DTN environments or the traces which are built synthetically. The network parameters (number of nodes etc.) are set according to the traces used.

5.1. Algorithms in comparison

We compared existing DTN algorithms with their CIT-using modified versions. First, we compared Shortest Path Routing (SPR) with Conditional Shortest Path Routing (CSPR) which is described in Section 4.1.3. Then, we compared the existing and revised versions of three metric-based DTN routing algorithms: Prophet [2], Fresh [29] and SimBet [24]. In the revised versions of these algorithms (referred to as C-Prophet, C-Fresh and C-SimBet to underline that they use CIT), A forwards the message to B if $\tau_A(D|B) > \tau_B(D|A)$ is also satisfied (in addition to the algorithm's own forwarding condition). In the graphs, we also give the results obtained by Epidemic Routing [6] since it achieves the optimum delivery ratio and delay (at high cost, however).

5.2. Datasets

For the main simulations, we used three real and one synthetic DTN traces. Real traces are from RollerNet [15], Cambridge [19] and Haggle [20] datasets where Bluetooth sightings between respectively 62, 36 and 41 user mobile devices are recorded. Further details of these traces can be found in the crawdad archive [31]. Synthetic traces are generated using a community-based mobility model which is similar to the models in [25,32,33]. In a 1000 units by 1000 units square region, we generated N_c randomly located non-overlapping community regions (home, work, school etc.) of size 100 units by 100 units and distributed N_p nodes (i.e. people) to these community regions. For each node, we randomly assigned V communities to visit (i.e. commonly visited places for a person in a day). Each node first selects a random point within the next community region in its list, assigns a random speed in range $[V_{min}, V_{max}]$ and moves towards the target point with that speed. Once it reaches that point, it randomly assigns a visit duration in range $[T_{min}, T_{max}]$ and randomly walks within the community region for that visit duration. Once that duration expires, it moves to the next community in its list in a similar way. Each node visits all the communities in its list as indicated, then once all of them are done (i.e. the end of day), they again start the same process and start visiting the communities in their list. While nodes are moving, we record the meetings between nodes assuming they have a transmission range of R. The default values for the parameters are $N_c = 10$, $N_p = 50$, $V = 5$, $V_{min} = 10$ units, $V_{max} = 50$ units, $T_{min} = 20$ time units, $T_{max} = 50$ time units. However, we also looked at the effects of different values of parameters in simulations. We also used large scale WiFi traces [34] to evaluate the performance of the proposed approach in large scale networks.

5.3. Simulation results

To collect several routing statistics, we have generated traffic on the aforementioned traces. For each simulation run, after a warm up period[4] (20% of the data), we generated 5000 messages from a random source node to a random destination node at each t seconds. In RollerNet, since the duration of the experiment is short, we set $t = 1$ s, but for Cambridge and Haggle datasets, we set $t = 1$ min and $t = 30$ s, respectively. For synthetic traces, we set $t = 10$ time units. Besides this single difference, we compare all algorithms in the same conditions.

For main simulations, we assume that the nodes have enough buffer space to store every message they receive,[5] the bandwidth is high and the contact durations of nodes are long enough to allow the exchange of all messages between nodes. These assumptions are reasonable in view of the capabilities of today's technology and are also used commonly in previous studies [35,26]. Any change in the current assumptions is expected to affect the performance of compared algorithms in the same way since they use one copy of the message. Moreover, we used a simplified slotted CSMA MAC model as in [7]. We ran each simulation 10 times with different seeds and in each run, we collect statistics by running each algorithm on the same set of messages. All results plotted in the figures show the averages of results obtained in all runs.

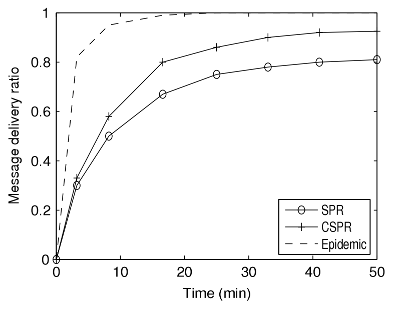
5.3.1. Comparison of CSPR and SPR

Fig. 5(a) shows[6] the delivery ratios achieved in CSPR and SPR algorithms with respect to time (i.e., TTL of messages) in RollerNet traces. Clearly, the CSPR algorithm delivers more messages to their destinations than the SPR algorithm. Moreover, it achieves lower average delivery delay than the SPR algorithm. For example, CSPR delivers 80% of all messages after 17 min with an average delay of almost 6 min, while SPR achieves the same delivery ratio only after 41 min and with an average

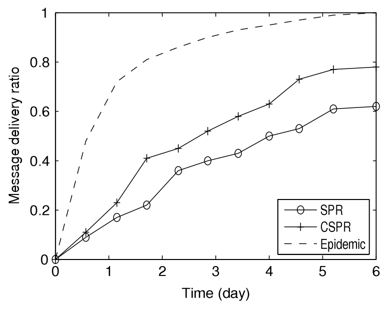
[4] During the warm up period, nodes build some encounter history to compute their initial CIT values. After the warm up period, as the messages are received and new meetings happen, CIT values are updated in parallel and the forwarding decisions are performed using the updated CIT values.

[5] The largest number of messages in a node buffer over all simulations was 120 messages, so in the order of 12 Mb, well below the buffer space in modern wireless devices.

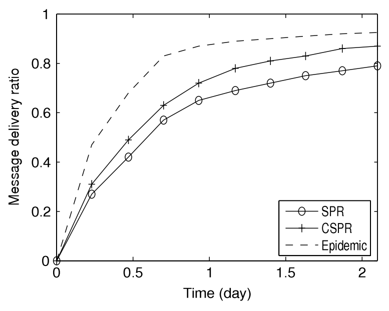
[6] Error bars are not shown since they are so small.



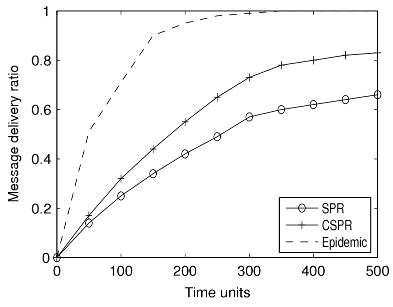
(a) RollerNet traces.



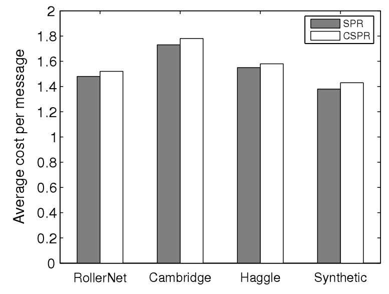
(b) Cambridge traces.



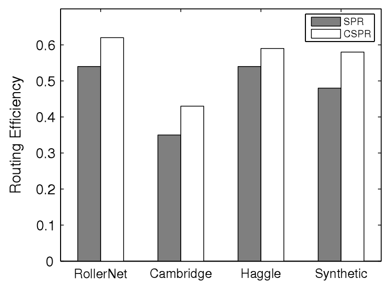
(c) Haggle traces.



(d) Synthetic traces.

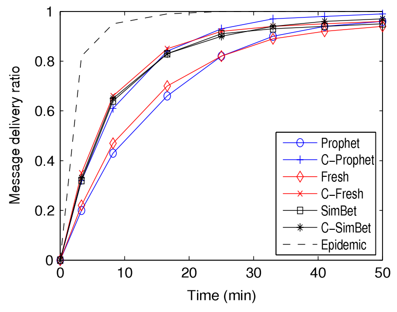


(e) Average cost.

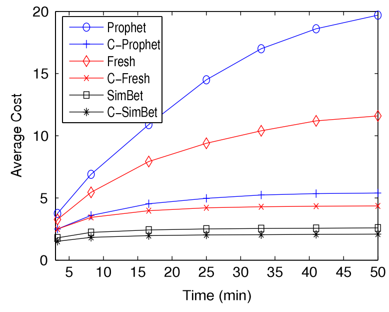


(f) Routing efficiency.

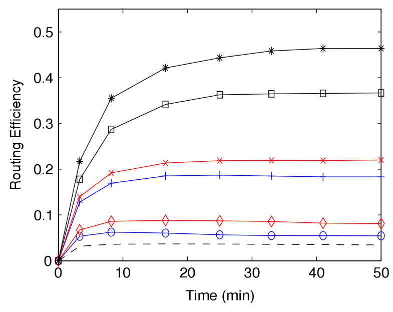
Fig. 5. Comparison of SPR and CSPR: message delivery ratio (a–d), cost (e) and routing efficiency (f).



(a) Delivery ratio.



(b) Average cost.



(c) Routing efficiency.

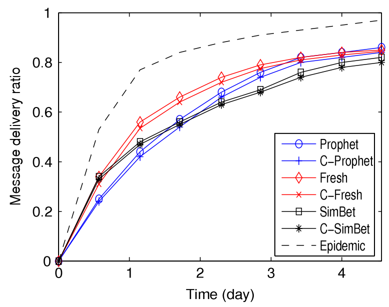
Fig. 6. Comparison using RollerNet traces.

delay of 12 min. Moreover, as it is shown in Fig. 5(e), average costs in SPR and CSPR are very close (1.48 and 1.52 respectively) to each other (and much smaller than the average cost in epidemic routing which is around 25). We also observe better delivery ratios achieved by the CSPR algorithm in Cambridge and Haggle traces in Fig. 5(b) and (c), respectively. In Cambridge traces, after 6 days, CSPR delivers 78% of all messages with an average delay of 2.6 days, however SPR can only deliver 62% of all messages to their destination with an average delay of 3.2 days. Moreover, average costs in SPR and CSPR are 1.73 and 1.78 respectively while it is around 16 in epidemic routing. Similarly, in Haggle traces, with an average cost close to each other, CSPR delivers 87% of all messages by the end of the simulation whereas SPR can only achieve 78% delivery ratio. The results with synthetic data in Fig. 5(d) also support the results based on real traces. While SPR delivers 65% of messages, CSPR delivers 82% of them when TTL of messages is set to 500 time units.

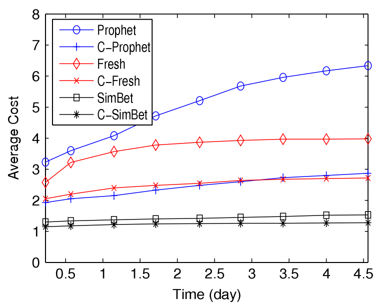
Fig. 5(f) compares the routing efficiency [36] of SPR and CSPR in all four traces. It shows an increase of 10%–22% in routing efficiency with the usage of CIT. However, if we compare the percentage of undelivered messages in these algorithms that are delivered in Epidemic routing, we can observe a higher performance increase. For example in Haggle traces, Epidemic routing delivered 94% of all messages. CSPR lost only 7% of these messages, while SPR lost 16% of them. Hence, CSPR achieved over 55% improvement over SPR.

5.3.2. Evaluation of modified metric-based algorithms

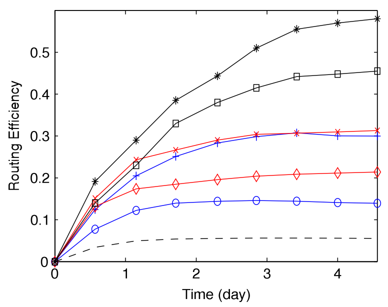
In Fig. 6(a), we show the delivery ratios achieved in RollerNet traces. Clearly, C-Prophet and C-Fresh provide a higher delivery ratio than their original versions but C-SimBet achieves a similar delivery ratio as SimBet. Moreover, as Fig. 6(b)



(a) Delivery ratio.

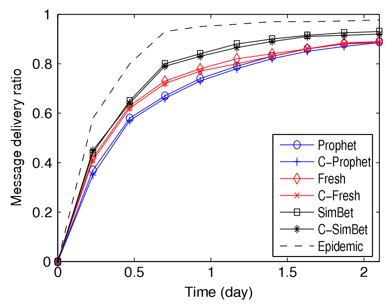


(b) Average cost.

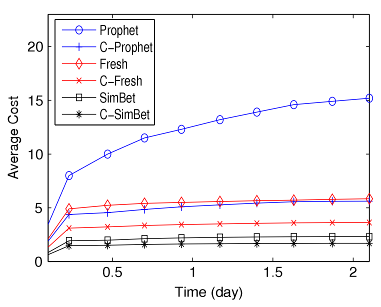


(c) Routing efficiency.

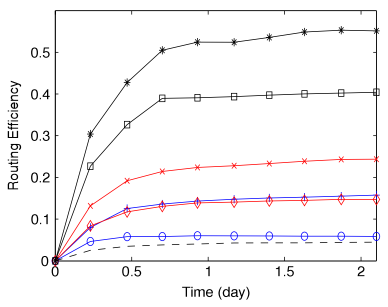
Fig. 7. Comparison using Cambridge traces.



(a) Delivery ratio.

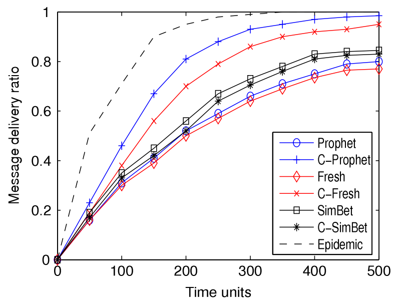


(b) Average cost.

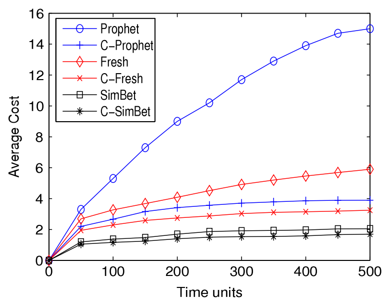


(c) Routing efficiency.

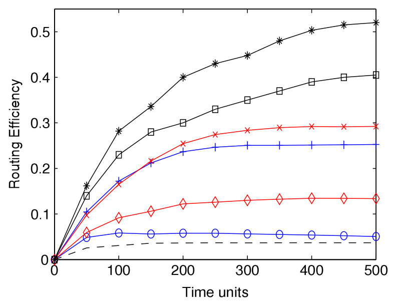
Fig. 8. Comparison using Haggle traces.



(a) Delivery ratio.



(b) Average cost.

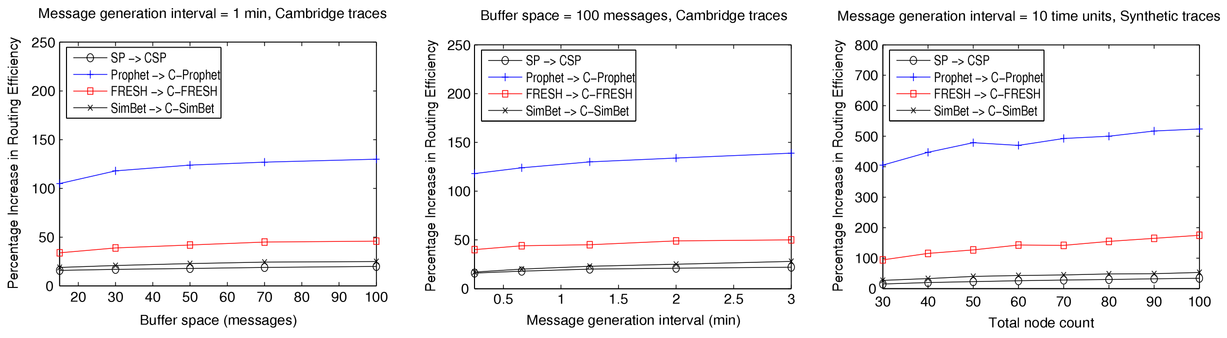


(c) Routing efficiency.

Fig. 9. Comparison using synthetic traces.

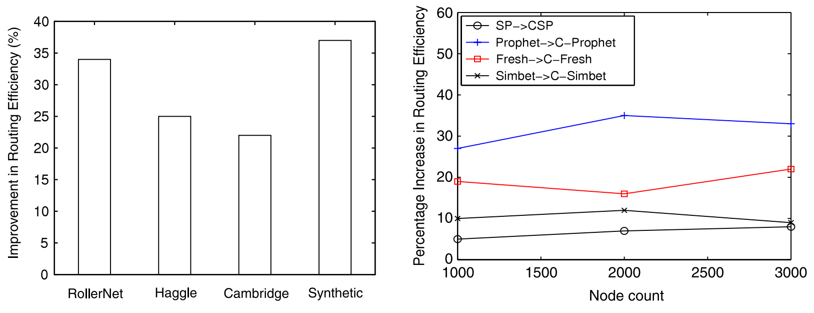
shows, average cost is lower for the modified algorithms. For example, C-Prophet delivers 90% of all messages after 23 min with average delay of 7.8 min and average cost of 4.83 hops. However, the original Prophet reaches the same delivery ratio only after 33 min with average delay of 13.5 min and average cost of 17.02. A similar situation is also observed between C-Fresh and Fresh, and C-SimBet and SimBet. As a result, over 100% increase in C-Prophet and C-Fresh, and around 30% increase in C-SimBet is achieved in routing efficiency (Fig. 6(c)).

When we look at the results obtained from Cambridge and Haggle traces in Figs. 7 and 8, we observe a different improvement. As it is seen in Figs. 7(a) and 8(a), revised and original versions of all algorithms have similar delivery ratios (and therefore similar average delays). However, as Figs. 7(b) and 8(b) show, average costs in modified versions are lower than they are in the original ones. This shows that when CIT is used as an additional delivery metric, the nodes choose better next hops so that the cost decreases while still keeping the original delivery ratio. Therefore, again the routing efficiency (Figs. 7(c) and 8(c)) is increased in all revised algorithms remarkably. The results with synthetic data in Fig. 9 also demonstrate the superiority of the revised algorithms. More (in C-Prophet and C-Fresh) or at least the same number of messages (in C-SimBet) are delivered with lower cost when compared to the original algorithms. From the above results, we clearly observe the benefit of CIT in metric-based forwarding algorithms.



(a) Effect of buffer space. (b) Effect of message generation interval. (c) Effect of node count.

Fig. 10. Extensive results.



(a) Improvement over CREST. (b) Results with WiFi traces.

Fig. 11. Results showing (a) the comparison with CREST and (b) the performance in WiFi traces.

5.3.3. *Effects of simulation parameters on results*

We evaluate here the impact of some parameters on the results. First, we look at the scenarios where the buffer space at nodes is limited. Assuming that nodes use the FIFO buffer management scheme, we measured routing efficiency improvements delivered by the proposed metric over the original algorithms. Fig. 10(a) shows the results for different buffer sizes in the range of [15–100] messages (in Cambridge traces). For these simulations we kept the message generation interval $t = 1$ min and $TTL = 4$ days. The results show that in the modified versions of algorithms, the increase in the routing efficiency grows as the buffer space increases. Moreover, the increase converges to a constant value after sufficient buffer spaces is allocated. CSPR, C-SimBet, C-Fresh, and C-Prophet offer 22%, 26% 48% and 130% increase in the routing efficiency over their original algorithms, respectively.

In Fig. 10(b), we observe similar results with different message generation intervals. As the messages are generated more frequently, due to buffer overflow, some messages are lost. However, the routing efficiency of algorithms is still remarkably increased with the modified versions. Finally, we changed the node count in the network (in synthetic traces) and looked at the effect of node count on results. Fig. 10(c) clearly shows that the increase in routing efficiency rises as the node count increases. This is because in synthetic data, temporal correlation between the meetings of nodes increases due to the higher number of nodes in each community. Thus, CIT provides more accurate information about node relations.

5.3.4. *Comparison with closest related work*

Even though several DTN routing algorithms have been proposed in literature, they usually assume that the meetings of a node with other nodes are independent and identically distributed. The closest study to our work is in [14], where a new metric, *conditional residual time* (CREST), which computes the remaining time for the meetings of two nodes based on the condition that t time units has passed since their last encounter, is proposed. However, the relations with other nodes is still not considered in this computation and meetings of a node with other nodes are assumed independent. In case of temporal correlation between the meetings of a node with other nodes, CIT can predict the remaining time to the next meeting of nodes more accurately, while CREST cannot differentiate the cases where two different nodes are met at the same elapsed time since the last meeting with destination and yields less accurate prediction. Fig. 11(a) shows how much improvement CIT can achieve over CREST while maintaining (sometimes increasing) the delivery rate in all traces. The results clearly show the superiority of CIT over CREST with an improvement in the range of 20%–37%.

5.3.5. Scalability of proposed approach

We also evaluated the performance of the proposed approach in a large scale network. Due to a lack of real DTN traces with many nodes, we used a large scale dataset of WiFi connection traces [34]. The traces contain 587,782 user sessions for 69,689 (distinct) users, which were collected from 206 hotspots for three years. We assumed that the users that are connected to the same hotspot can communicate with each other to obtain the meetings (i.e. communication opportunity) of nodes similar to the DTN environment. There are many users which appear a few times in the dataset, thus their meeting counts with other nodes are very small. Therefore, we first identified the top 1000, 2000 and 3000 nodes having the most meetings and utilized the meeting history that occurs among these nodes for the evaluation of the proposed approach. As Fig. 11(b) shows, the modified versions of algorithms have much better routing efficiency (with similar or higher delivery rate) compared to the original algorithms. The increase achieved via modified algorithms is smaller than the increase in routing efficiency shown in the results with other datasets. This can be due to the difference of WiFi traces than DTN traces, however, the results show that the proposed approach can provide enough improvement even in large scale networks. CSPR, C-SimBet, C-Fresh, and C-Prophet algorithms offer 8%, 10% 17% and 30% increase on the average in the routing efficiency over their original algorithms, respectively.

6. Conclusion and future work

In this paper, we focused on the routing problem in delay tolerant networks (DTN). First, we introduced a new metric called conditional intermeeting time (CIT) which is the average time that passes from the time a node meets with a neighbor until the time it meets another one. Next, we presented an analysis of this metric showing why it can improve representation accuracy of node relations. Then, we looked at the effects of this metric on existing DTN routing algorithms. To this end, we modified their current designs to enable them to use CIT. Finally, through extensive simulations based on both the real DTN traces and synthetic mobility traces, we evaluated the modified algorithms and demonstrated their superiority over the original ones.

In our future work, we plan to extend the definition of CIT to include more than one meeting in the contact history. To achieve this, we plan to use probabilistic context free grammars (PCFG) and the construction algorithm presented in [37].

References

[1] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Efficient routing in intermittently connected mobile networks: The single-copy case, IEEE/ACM Transactions on Networking 16 (1) (2008).

[2] A. Lindgren, A. Doria, O. Schelen, Probabilistic routing in intermittently connected networks, SIGMOBILE Mobile Computing and Communication Review 7 (3) (2003).

[3] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, MaxProp: routing for vehicle-based disruption- tolerant networks, in: Proc. IEEE Infocom, April 2006.

[4] C. Mascolo, M. Musolesi, CAR: context-aware adaptive routing for delay tolerant mobile networks, IEEE Transactions on Mobile Computing 8 (2) (2009) 246–260.

[5] E. Bulut, B.K. Szymanski, Exploiting friendship relations for efficient routing in mobile social networks, IEEE Transactions on Parallel and Distributed Systems 23 (12) (2012) 2254–2265.

[6] A. Vahdat, D. Becker, Epidemic routing for partially connected ad hoc networks, Duke University, Tech. Rep. CS-200006, 2000.

[7] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Efficient routing in intermittently connected mobile networks: the multi-copy case, IEEE/ACM Transactions on Networking (2008).

[8] A. Balasubramanian, B.N. Levine, A. Venkataramani, Replication routing in DTNs: a resource allocation approach, IEEE Transactions on Networking 18 (2) (2010).

[9] Y. Wang, S. Jain, M. Martonosi, K. Fall, Erasure coding based routing for opportunistic networks, in: Proc. of ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN), 2005.

[10] E. Bulut, Z. Wang, B.K. Szymanski, Cost efficient erasure coding based routing in delay tolerant networks, in: Proc. of the ICC, Capetown, South Africa, May 2010.

[11] I. Psaras, L. Wood, R. Tafazolli, Delay-/disruption-tolerant networking: state of the art and future challenges, Technical Report, University of Surrey, UK, 2009.

[12] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, J. Scott, Impact of human mobility on the design of opportunistic forwarding algorithms, in: Proc. of INFOCOM, 2006.

[13] X. Zhang, J.F. Kurose, B. Levine, D. Towsley, H. Zhang, Study of a bus-based disruption tolerant network: mobility modeling and impact on routing, in: Proc. of ACM MobiCom, 2007.

[14] S. Srinivasa, S. Krishnamurthy, CREST: an opportunistic forwarding protocol based on conditional residual time, in: Proc. of SECON, 2009.

[15] P.U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Amorim, J. Whitbeck, The accordion phenomenon: analysis, characterization, and impact on DTN routing, in: Proc. of Infocom, 2009.

[16] P. Hui, J. Crowcroft, E. Yoneki, BUBBLE rap: social based forwarding in delay tolerant networks, in: Proc. of ACM MobiHoc, 2008.

[17] C. Liu, J. Wu, Practical routing in a cyclic MobiSpace, IEEE/ACM Transactions on Networking 19 (2) (2011).

[18] E. Bulut, S. Geyik, B. Szymanski, Efficient routing in delay tolerant networks with correlated node mobility, in: Proc. of MASS, Nov, 2010.

[19] J. Leguay, A. Lindgren, J. Scott, T. Friedman, J. Crowcroft, P. Hui, CRAWDAD data set upmc/content (v. 2006-11-17), 2006. Downloaded from http://crawdad.cs.dartmouth.edu.

[20] A European Union funded project in Situated and Autonomic Communications. www.haggleproject.org.

[21] N. Eagle, A. Pentland, D. Lazer, Inferring social network structure using mobile phone data, Proceedings of the National Academy of Sciences 106 (36) (2009) 15274–15278.

[22] S. Jain, K. Fall, R. Patra, Routing in a delay tolerant network, in: Proc.of ACM SIGCOMM, Aug. 2004.

[23] E.P.C. Jones, L. Li, P.A.S. Ward, Practical routing in delay tolerant networks, in: Proc. of ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN), 2005.

[24] E. Daly, M. Haahr, Social network analysis for information flow in disconnected delay-tolerant MANETs, IEEE Transactions on Mobile Computing 8 (5) (2009).

[25] F. Li, J. Wu, LocalCom: a community-based epidemic forwarding scheme in disruption-tolerant networks, in: Proc. of IEEE Secon 2009.

[26] E. Bulut, Z. Wang, B. Szymanski, Cost-effective multi-period spraying for routing in delay tolerant networks, IEEE/ACM Transactions on Networking 18 (5) (2010).

[27] EasyFit: distribution fitting software. http://www.mathwave.com/ (last accessed in August, 2013).

[28] D. Bertsekas, R. Gallager, Data Networks, second ed., 1992.

[29] H. Dubois-Ferriere, M. Grossglauser, M. Vetterli, Age matters: efficient route discovery in mobile ad hoc networks using encounter ages, in: Proc. of ACM MobiHoc, 2003.

[30] T. Spyropoulos, K. Psounis, C. Raghavendra, Spray and focus: efficient mobility-assisted routing for heterogeneous and correlated mobility, in: Proc. of IEEE PerCom, 2007.

[31] CRAWDAD data set. http://crawdad.cs.dartmouth.edu.

[32] W. Hsu, T. Spyropoulos, K. Psounis, A. Helmy, Modeling time variant user mobility in wireless mobile networks, in: IEEE INFOCOM, 2007.

[33] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Performance analysis of mobility-assisted routing, in: Proc. of MobiHoc, 2006.

[34] M. Lenczner, B. Grégoire, F. Proulx, CRAWDAD trace set ilesansfil/wifidog/session (v. 2007-08-27), 2013. Downloaded from http://crawdad.cs.dartmouth.edu/ilesansfil/wifidog/session. July.

[35] C. Liu, J. Wu, On multicopy opportunistic forwarding protocols in nondeterministic delay tolerant networks, IEEE Transactions on Parallel and Distributed Systems 23 (6) (2012) 1121–1128.

[36] J.M. Pujol, A.L. Toledo, P. Rodriguez, Fair routing in delay tolerant networks, in: Proc. of Infocom, 2009.

[37] S. Geyik, E. Bulut, B. Szymanski, Grammatical inference for modeling mobility patterns in networks, IEEE Transactions on Mobile Computing (TMC) 12 (11) (2013) 2119–2131.