## APPENDIX A
## "OUT-POOL" CASE

The "out-pool" case contains tracking foreign nodes algorithm and tracking missing nodes algorithm.

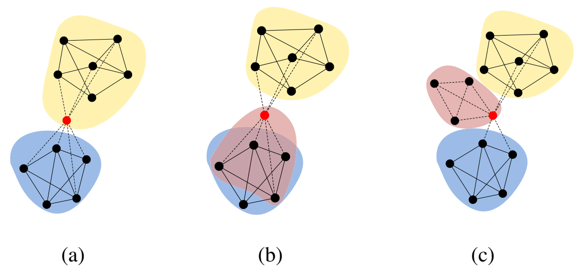

Fig. 6: The red node is a foreign node with adding edges depicted by dotted lines. In (a), the foreign node joins its adjacent community. In (b), the foreign node forms new communities with neighbors. In (c), the foreign node unites the solitary nodes to form a new community.

**Algorithm 4** Tracking Foreign Nodes

```
Input: the current community structure $\mathcal{C}_t$
Output: the updated structure $\mathcal{C}_{t+1}$
 1: if node $u$ is added without edges, then
 2:   $CS^t = CS^t \bigcup \{u\}$
 3: else $u$ with edges
 4:   $x_t \leftarrow$ apply Definition 4 on the community graph of $\mathcal{C}_t$
 5:   for each $w_{uv}^t \in W_t$
 6:     if $w_{uv}^t < x_t$
 7:       $E_t \leftarrow E_t \setminus (u, v)$
 8:   update the set of $N_t(u)$
 9:   $C_1^t, C_2^t, ..., C_k^t \leftarrow$ adjacent communities of $u$
10:   for $i = 1$ do to $k$
11:     $O_t(u, v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on $C_i^t \bigcup \{u\}$
12:     if $\Phi(O_t(u, v)) \geq \delta(O_t(u, v))$ and $|V_t(u, v)| \geq 4$
13:       $C_i^t \leftarrow C_i^t \bigcup \{u\}$
14:     else
15:   $O_t(u, v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on $C_i^t \bigcap N_t(u)$
16:       if $\Phi(O_t(u, v)) \geq \delta(O_t(u, v))$ and $|V_t(u, v)| \geq 4$
17:         define $V_t(u, v)$ of $O_t(u, v)$ as a new community $C'$
18:   for $v \in CS^t$ and $Com_t(u) \bigcap Com_t(v) = \emptyset$
19:     $O_t(u, v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on
20:                  $N_t(u) \bigcap N_t(v)$
21:       if $\Phi(O_t(u, v)) \geq \delta(O_t(u, v))$ and $|V_t(u, v)| \geq 4$
22:       define $V_t(u, v)$ of $O_t(u, v)$ as a new community $C'$
23: merge overlapping communities on $C_1^t, C_2^t, ..., C_k^t$ and $C'$
24: update $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$
```

Firstly, we analyze Algorithm 4 about adding foreign nodes case. There are two possibilities, one is the node added without edges, the other is added with edges. If node $u$ satisfies the former case, we simply join $u$ to the current community structure. If $u$ is the latter case, it becomes a little complicated and needs three operations, as illustrated in Fig. 6: 1) Because $u$ is added with edges, it may join to its adjacent communities, *i.e.*, step $9-13$. 2) Uniting its neighbors, the foreign node $u$ may form new communities, *i.e.*, step $15-17$. 3) Considering the set of solitary nodes, node $u$ may shape new communities, *i.e.*, step $18-22$.

Secondly, we study Algorithm 5 about tracking missing nodes case. 1) If node $u$ is a solitary node or $d_u = 1$, we simply remove the node from the current community structure. 2) Otherwise, there are two operations, as illustrated in Fig. 7. One is the remaining structure can maintain the original community, *i.e.*, step $8-12$, the other is the remains may form new communities, *i.e.*, step $14-16$.

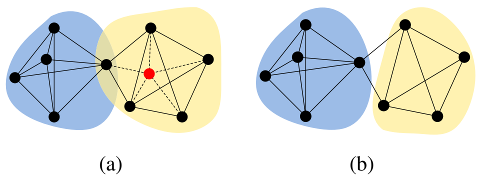

Fig. 7: The red node represents the missing node with removing edges depicted by dotted lines. In (a), the remaining structure can maintain the original shape. In (b), the remains forms two new communities.

**Algorithm 5** Tracking Missing Nodes

```
Input: the current community structure $\mathcal{C}_t$
Output: the updated structure $\mathcal{C}_{t+1}$
 1: if $u$ is a solitary node or $d_u^t = 1$
 2:   $\mathcal{C}_t \leftarrow \mathcal{C}_t \setminus \mathcal{C}_t(u)$
 3: else
 4:   $x_t \leftarrow$ apply Definition 4 on the community graph of $\mathcal{C}_t$
 5:   for each $w_{uv}^t \in W_t$
 6:     if $w_{uv}^t < x_t$
 7:       $E_t \leftarrow E_t \setminus (u, v)$
 8:   for each subset $C_i^t$ in $\mathcal{C}_t(u)$ or in $\mathcal{C}_t(v)$
 9:     $O_t(u, v) \leftarrow$ the induced subgraph of the filter graph $G_t(x_t)$
10:              based on the remaining nodes in one subset $C_i^t$ of $\mathcal{C}_t(u)$
11:      if $\Phi(O_t(u, v)) \geq \delta(O_t(u, v))$ and $|V_t(u, v)| \geq 4$
12:        $C_i^t \leftarrow V_t(u, v)$
13:      else
14:        sort the weight of $E_t(u, v)$ in a descending order
15:        from the largest weighted edge $(u, v) \in E_t(u, v)$
16:   do Algorithm 2 step $8-10$ to gain new communities sequence $C'$
17: merge overlapping communities
18: update $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$
```

## APPENDIX B
## "IN-POOL" CASE

The "in-pool" case contains tracking adding edges algorithm and tracking removing edges algorithm.

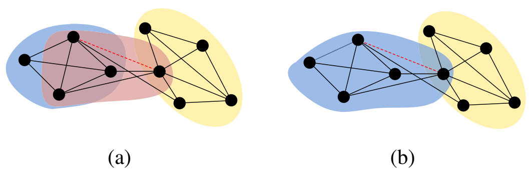

Fig. 8: The red dotted line represents an adding edge. In (a), the new edge shapes a new community. In (b), for an adding edge, one of its endpoints joins the community of the opposite side.

Firstly, we discuss Algorithm 6 about adding edges case. There are two possibilities, one is two endpoints of the adding edge are in the same community, the other is in the different communities. In the former case, community structure does not change, because adding edges increases the weighted density of communities. In the latter case, we further divide it into two operations, as illustrated in Fig. 8. 1) If the adding edges come from current nodes, we decide whether the edge $(u, v)$ can form a new community, *i.e.*, step $4-9$. Besides, we still need to judge whether the node

**Algorithm 6** Tracking Adding Edges

```
Input: the current community structure $\mathcal{C}_t$
Output: the updated structure $\mathcal{C}_{t+1}$
 1: if $Com_t(u) \bigcap Com_t(v) \neq \emptyset$
 2:   $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t$
 3: else
 4:   if $Com_t(u) \neq \emptyset$ and $Com_t(v) \neq \emptyset$
 5:     if $Com_t(u) \bigcap Com_t(v) = \emptyset$ then
 6:       $O_t(u,v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on
 7:                $N_t(u) \bigcap N_t(v)$
 8:       if $\Phi(O_t(u,v)) \geq \delta(O_t(u,v))$ and $|V_t(u,v)| \geq 4$
 9:         define $V_t(u,v)$ of $O_t(u,v)$ as a new community $C'$
10:       else
11:         for each subset $C_i^t$ in $\mathcal{C}_t(u)$ or in $\mathcal{C}_t(v)$
12:           $O_t(u,v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on one
13:              subset $C_i^t$ of $\mathcal{C}_t(u) \cup \{v\}$ or one subset $C_i^t$ of $\mathcal{C}_t(v) \cup \{u\}$
14:           if $\Phi(O_t(u,v)) \geq \delta(O_t(u,v))$ and $|V_t(u,v)| \geq 4$
15:             $C_i^t \leftarrow C_i^t \cup \{v\}$ or $C_i^t \leftarrow C_i^t \cup \{u\}$
16:   if $(Com_t(u) = \emptyset$ and $Com_t(v) \neq \emptyset)$ or $(Com_t(v) = \emptyset$ and
17:       $Com_t(u) \neq \emptyset)$
18:     only do Algorithm 6 step $5-9$
19: merge overlapping communities
20: update $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$
```

$u$ or $v$ will join the community of the opposite side, *i.e.*, step $10-15$. 2) If the adding edges come from the new foreign nodes, we only need to process the edge $(u,v)$, *i.e.*, judging whether to shape a new community or not, described in step $16-18$. Some operations about two endpoints have been done in Algorithm 4.

Secondly, we study Algorithm 7 about tracking removing edges case. There are also two possibilities, one is two endpoints of the removing edge are in the different community, the other is in the same communities. In the former case, the community structure does not change. In the latter case, as illustrated in Fig. 9, we only need to concern the case that the removing edges come from the existing networks. Because if the removing edges come from the missing nodes, the corresponding operations have been done in Algorithm 5. Therefore, in Algorithm 7, 1) We decide whether the remaining structure can still maintain or not, *i.e.*, step $2-6$. 2) Otherwise, the remains can form some new communities, *i.e.*, step $8-10$.

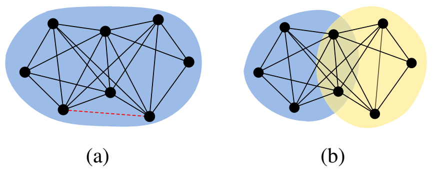

Fig. 9: The red dotted line represents a removing edge. In (a), the remaining structure can still maintain. In (b), the remaining structure forms two new communities.

## Appendix C Implementation of Community Detection Method SAWD

For algorithm 2 and 3 are centralized. For algorithm 2, it only executes once at the initial constructing stage. For algorithm 3, it is only used to collect edge weights and calculate the media of the weight set $W_t$ for finding the

**Algorithm 7** Tracking Removing Edges

```
Input: the current community structure $\mathcal{C}_t$
Output: the updated structure $\mathcal{C}_{t+1}$
 1:   if $Com_t(u) \neq \emptyset$ and $Com_t(v) \neq \emptyset$
 2:   for each subset $C_i^t$ in $\mathcal{C}_t(u)$ or in $\mathcal{C}_t(v)$
 3:    $O_t(u,v) \leftarrow$ the induced subgraph of $G_t(x_t)$ based on the
 4:    remaining nodes in one subset $C_i^t$ of $\mathcal{C}_t(u)$ after removing $(u,v)$
 5:     if $\Phi(O_t(u,v)) \geq \delta(O_t(u,v))$ and $|V_t(u,v)| \geq 4$
 6:       $C_i^t \leftarrow V_t(u,v)$
 7:     else
 8:       sort the weight in $E_t(u,v)$ in a descending order
 9:       from the largest weighted edge $(u,v) \in E_t(u,v)$
10:   do Algorithm 2 step $8-10$ to gain new communities sequence $C'$
11:  merge overlapping communities
12:  update $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$
```

changed edges as time goes by. These costs are bearable for a global centralized server and all these work can be done effectively.

For the dynamic tracking algorithm 4-7, they all use local structure information to handle all changes, thus, a decentralized deployed way is fit for them. That is to say, nodes undertake a large portion of work. A node has perfect knowledge of its neighbors and some local approximation knowledge captured by its neighbors. Some required information is transferred through node to node, like literature [32]. Node $u$ has the knowledge of its neighbors $N_t(u)$ and its belonging community(communities) $Com_t(u)$ (In practice, each community label in $Com_t(u)$ is marked with the set of its members). Through one of its neighbors node $v$, node $u$ also can gain the information of $N_t(v)$ and $Com_t(v)$. Based on these, the knowledge of $O_t(u,v)$ can be obtained by node $u$. Besides, node $u$ can collect the information from its neighbors and some local approximation knowledge captured by its neighbors. For example, a community contains node 1, node 3, node 5 and node 7. For node 1, its neighbors are node 3 and node 5. Because, node 1 has the knowledge of $N_t(1)$ and $Com_t(1)$. Thus, it knows its neighbors node 3 and node 5 and all its community members. Then, node 1 can collect the connectivity information from its neighbors and some local connectivity knowledge captured by its neighbors. Because the weighted criterion of community in Definition 6 can limit the emergence of large communities and the communities are located geographically, so, node 1 can know the sum of edge weight between any two nodes in one of its belonging community $C_i^t$ and can judge the local community for each dynamic change. Besides, each node only collects the information of its own community(communities), not the entire network, thus, the work load of a node will not be too large to bear.

The efficient distributed implementation is a common issue and indeed not an easy problem in community detection research. Usually, efficient distribution and precise detection result cannot be gained simultaneously. In the future, a hybrid underlying infrastructure will be a solution for this problem.

## Appendix D
## Costs of Community Detection Method SAWD

**Lemma 1.** *The time complexity of Algorithm 2 is $O(M + M\log M + N^2)$.*

*Proof:* Assuming there are $N$ nodes and $M$ edges in a social weighted graph. First, time complexity of getting the median of the set of weights $W_t$ is $O(M)$. Next, because there are $M$ edges which require to compare with the median, the time complexity of comparison is $O(M)$. Then, for Step 6, time complexity of sorting the weighted edges is $O(M\log M)$. Finally, from Step 7 to Step 12, we have to find the intersection of $N_t(u)$ and $N_t(v)$. Because $|N_t(u)| + |N_t(v)| = d^t(u) + d^t(v)$, the time complexity for each weighted edge is $\Sigma_{u\in V_t} d^t(u) = 2M$. From Step 13 to Step 17, suppose there are $N_0$ raw communities in $\mathcal{C}_{raw}$ at Step 12, according to Lemma 11.8 in [33], when the number of nodes in the intersection of any two communities is upper bounded by a constant $\alpha$, the number of raw communities $N_0$ is $O(N)$, so the time complexity of combining is $O(N^2)$. Therefore, the total time complexity of Algorithm 2 is $O(M + M\log M + N^2)$. □

**Lemma 2.** *The time complexity of Algorithm 3 is $O(M)$.*

*Proof:* First, time complexity of getting the median of the set of weights is $O(M)$. Next, because there are $M$ edges which require to compare with the median, the time complexity of comparisons is $O(M)$. Finally, the time complexity of getting the difference set of the two edge sets at time slot $t-1$ and $t$ is $O(M)$. Therefore, the total time complexity of Algorithm 3 is $O(M)$. □

For Algorithm 4-7, because they locally deal with the network changes (including judging a new community and combining the overlapped communities), the time complexity of them is upper bounded by Algorithm 2.

Note that, comparing with [7], the time complexity of Algorithm 2 is higher with order of $O(M\log M)$ because of sorting the weighted edges.

## Appendix E
## Detailed Comparisons and Our Improvement

### E.1 Comparisons

There exist some other methods and concepts [3], [4], [34], [35] which may be confused with the definition of our local activity and social similarity. We give the detailed explanations so as to distinguish them and show our improvement.

(1) In Simbet [3] and BUBBLE RAP [4], they use *betweenness centrality* in data forwarding. Betweenness measures the extent to which a node lies on the shortest paths linking other nodes. A node with a high betweenness centrality has a capacity of facilitating interactions between the nodes that it links. However, there exist some problems.

• Betweenness centrality used in them is applied in unweighted graphs, i.e., without considering contact frequency. In unweighted graphs, there will be an edge if there has a contact between two nodes at any time point $t$. But in reality, the contact probability may be too low to be utilized in data forwarding. That is to say, the relay may have no contact with others in future, so, this delivering will lead to an invalid transmission.

• Additionally, another problem in Simbet and BUBBLE RAP is using the global betweenness in entire or partial phase of data forwarding. In BUBBLE RAP and Simbet, the concept of community is explicitly or implicitly considered. Each node has its belonging community(ies), expect solitary nodes. If we deliver the message to a node having high global betweenness, although it indeed has high contact frequency with other nodes with respect to the entire network, it may be in a community which is irrelevant to or does not overlap with the destination community. On the contrary, the nodes which have similar communities with the destination, but have a little bit low global betweenness, can be good enough as relays.

(2) In SocialCast [34] and IRA [35], a common problem is that they use a form of simple weighted coefficient sum to combine the interests and connectivity as the routing guideline. This method makes the routing guideline (with weighted coefficient) rely on the types of datasets. It is not universal for all datasets and not convenient for popularity. Additionally, the weighted coefficient sum method is also too coarse and simple to reflect the combination relationship of the interests and degree connectivity. Besides, there exist other problems in SocialCast and IRA respectively.

• For SocialCast, it cannot deal with the overlapped interests. For example, if node $u$ has $a, b, c$ three kinds of interests, according to SocialCast, the nodes only with interest $a$ can be all chosen as relays. However, this raw method leads to large numbers of copies to result in network congestion and low delivery ratio. In practice, we require to find the nodes that have more similar interests with the destination.

• For IRA, the concept of community is proposed, however, it is just used in finding coordinator nodes and ambassador nodes when doing file searching and retrieval, and it is not used in designing the routing algorithm. The routing scheme in IRA is still similar with PROPHET [12], although IRA considers file interests. They simply use the encounter history to predict the future delivery probability. This kind of algorithms is not as good as community-based prediction algorithms in mobile social networks. Because in social networks, nodes belonging to the same community indicates they are more likely to meet each other. To some extent, it can reflect social preference. So, community structure is good at delivery prediction. The similar demonstration appears in literature [4]. In addition, in our data forwarding experiments (Section 5.3), we can see, Bubble Rap and Nguyen's Routing (community-based algorithms in Fig.5 (d)) perform better than PROPHET (noncommunity-based algorithms in Fig.5 (a)) in terms of delivery ratio.

### E.2 Our Improvement

Our method is different with aforementioned methods in two aspects. First, in our paper, we concern about the contact frequency. Our weighted social graphs are formed according to *contact frequency* among nodes in Section 2.1 and Section 5.2. Then, when doing *community detection*, we use communication critical value in Definition 4 to filter the low contact frequency nodes. And in routing algorithm LASS, our scheme is apt to choose the high *local activity* nodes to avoid the low contact frequency nodes. Thus, the improper relays with low contact frequency will not be chosen. Considering contact frequency to form a weighted social graph is one of essentials in our paper.

Second, we use the *inner product* of two forwarding utilities as *social similarity* to design a data forwarding scheme. In a forwarding utility, it includes the information of local activity and the number of communities/interests. We use the inner product method to choose a node that is more similar with the destination than the current message holder. The key point is that: the *similar local activity* in *similar communities* with the destination.

We assume that node $w$ is the destination node. There exists a unicast session from node $u$ to node $w$. The candidate relay is node $v$. If the social similarity between $v$ and $w$ is larger than $u$ and $w$, there will be two facts that can be proved. One, the candidate node $v$ has more common interests with the destination node $w$, *i.e.*, the number of non-zero vector component is large. The other, referring to the destination node $w$, the candidate node $v$ has high local activity values on the corresponding non-zero vector component. That is to say, in each vector component, node $v$ and $w$ are more anastomotic. Reflected in social networks, they are more similar in social aspects, *i.e.*, the interests groups and the local activity of node $v$ are proportionate to destination $w$. Intuitively, if the destination is in Rugby Club and University Chorus-two groups/communities, and the node local activity in Rugby Club is larger than in University Chorus. Then, the node that has the same characteristic with the destination is more appropriate as a relay than that has not. Our inner product method is universal for all kinds of datasets and do not have to identify the different parameters. Besides, the forwarding utility can handle a node with multiple interests. Therefore, LASS aims at choosing both high local activity and many common communities/interests with the destination simultaneously.

In Section 5.3, we do experiments to validate the advantage of our LASS, including comparisons with Epidemic, PROPHET, Simbet, BUBBLE RAP, Nguyens Routing (for SocialCast and IRA, we do not provide the comparisons, because the former cannot deal with the overlapped interests and the latters routing scheme is same with PROPHET).

## APPENDIX F DISCUSSION

We further explain several issues about our algorithm's designing and give some possible extensions for the future.

### F.1 DataSets Selection

There exist many collections of social networks datasets, such as CRAWDAD[5], Haggle iMotes[6] projects and Stanford SNAP Graph Library [7]. In above collections, Infocom 06 dataset, Sigcomm09 dataset, MIT Reality Mining dataset and Facebook dataset will be found. Based on them, some studies about relationship inference, behavior modeling and prediction, complex social studies, and information dissemination are carried out. These datasets can be classified into two kinds, one kind is the social friendship information, the other is the social proximity information. The former is about logical relationship, the latter is about geographic relationship. In our study, because we concern with the geographic encounter-based scenarios, the second kind of data (using Bluetooth discovery to gain proximity information) is appropriate for our experiments. In order to observe the social impact on data forwarding, we choose a long term observation-MIT Reality Mining Dataset as our experiment dataset. Our algorithm can also be applied to other datasets to validate it effectiveness.

### F.2 Choice about the Communication Critical Value

With continual adding and removing actions, the communication critical value $x_t$ is generated by calculating the median value of the weight set $W_t$ at each snapshot. This method can tackle both uniform and power-law distribution. However, it may have some more precise mathematical methods than ours to deal with the problem, which can be studied in the future.

### F.3 Edge Weight

In our study, the edge weight represents the encounter probability between two nodes. The edge weight may be concerned with mobility intensity, traffic interests or some other physical/logical social properties. But it is not in the scope of our research.

### F.4 Bounds for Combining Criterion of Communities

Like [7], we also use experimental method to gain the threshold for combining criterion of communities, *i.e.*, Definition 8. But in theory, we could give the upper and lower bound of it. The bounds meet two conditions, one is that the community structure should satisfy the weighted criterion of communities after combining, *i.e.*, Definition 6; the other is if two communities can combine with each other, we should try our best to do this. This theory bounds will be studied in the future.

5. http://crawdad.cs.dartmouth.edu/
6. http://www.haggleproject.org
7. http://snap.stanford.edu/data/index.html

### F.5 Efficient Way for Calculation of Local Activity

In our community detection method SAWD, we use both "communication critical value" (Definition 4) and "weighted criterion of communities" (Definition 6) to guarantee the relative high weighted edges to form meaningful communities. Then, under these chosen communities, according to the definition of local activity, a node with high local activity can mean that it can connect all nodes in the community tightly *in high probability*. However, there indeed exists a kind of extreme and rare case to make the definition of local activity look imperfect. For example, there is a node, say $u$, in a community C. It only meets another same community node, say $v$, extremely frequently (i.e., only one edge in community C is weighted high, the others are all weighted low). Then both node $u$ and $v$ have high local activity in community C. In this case, $u$ and $v$ will also be selected as a good relay for other nodes in community C. But actually they only meet each other frequently. Therefore, the current definition of local activity cannot show that it connects all nodes in the community tightly. In the future work, probably, when we define local activity, an upper bond on the weight between two nodes can further help alleviate the problem.

### F.6 Fairness Issues of LASS

In relative short time period (a session), some centrality-based social forwarding schemes indeed exist the problem that some popular nodes (large centrality) may be frequently used as relay nodes. These popular nodes may be abused and deplete the battery quickly, i.e., fairness issues of the forwarding scheme. However, in our paper, the popular node means a node having large social similarity with the destination. That is to say, the popular node is required to have not only a relative high local activity, but also more number of communities where the destination node is in. Therefore, comparing with precious centrality-based social forwarding scheme, LASS has alleviated the abuse of the popular nodes to some extent. We think that choosing the popular nodes and considering battery depletion together is worth to be studied in the future work.

## Appendix G
## Pros-and-cons of LASS

For pros-and-cons of LASS, we need to emphasize that our LASS is not the algorithm that uses long delay to gain high delivery ratio. From Fig.5 (b) and Fig.5 (e), we can see, if achieving the same number of successfully delivered messages, except Nguyen's Routing (we do not discuss it because our delay is only higher than it 6.7 percent), the other four comparison algorithms use far more relays than LASS. *Note that they are not in the same order. LASS is only of 10 order, but the comparison algorithms are of 1000 or 10000 order*. This is because LASS will not choose an improper relay and make large numbers of copies in relays. So, the main reason that leads to a little higher delay of LASS is the number of relays of LASS is far smaller than the other four comparison algorithms. It is not due to using long delay to gain high delivery ratio. Besides, through analysis of experimental data, from Fig.5 (c) and Fig.5 (f), we can gain, comparing with Epidemic, PROPHET, Simbet, BUBBLE RAP and Nguyen's Routing, the delay of LASS is only higher than them 28.2, 17.5, 25.6, 21.1, 6.7 percent on average respectively. Specially, from Fig.10 (a) and Fig.10 (b), for the same delivery ratio, except Epidemic (due to large numbers of copies), the delay of LASS is smaller than PROHPHET, Simbet, BUBBLE RAP and Nguyen's Routing. *Therefore, in conclusion, we use a little higher delay to achieve a prominent delivery ratio (some algorithms cannot achieve the delivery ratio that LASS achieves) and a far lower overhead.*

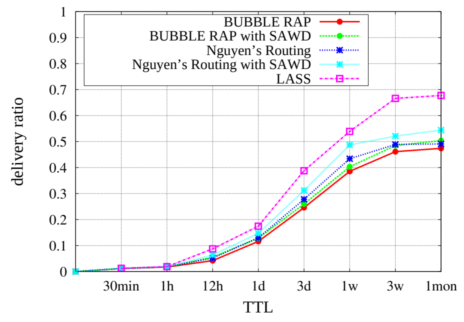

Fig. 12: Comparison results on BUBBLE RAP, Nguyen's Routing and LASS using SAWD detection method respectively.

## Appendix H
## Experiments for the Role of Local Activity and Social Similarity of LASS

In this section, especially, we use the same community detection method SAWD in algorithms BUBBLE RAP and Nguyen's Routing respectively to test their performance comparing with LASS. We want to verify, in our paper, besides the good community detection method, considering local activity and using social similarity to guide the routing path are also the important elements in improving data forwarding.

From Fig.12, we can see, in terms of delivery ratio, when using the same community detection method SAWD, both the performances of BUBBLE RAP and Nguyen's Routing are still poorer than LASS. LASS outperforms BUBBLE RAP and Nguyen's Routing with SAWD 37.47 and 23.24 percent respectively. Comparing with the results in Section 5.3.4 (the delivery ratio of LASS is higher than BUBBLE RAP with 46.18 percent, Nguyen's Routing with 34.64 percent on average), it validates that a good community detection can influence the data forwarding to some extent. Besides, it further verifies the important role of local activity and social similarity in data forwarding for MSNs.

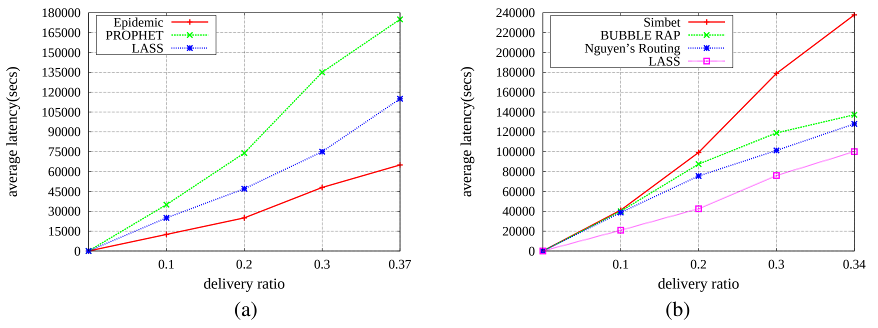

Fig. 10: Fig(a) and Fig(b) show the different average latency in different delivery ratio (achievable lower bound) for five comparison algorithms respectively.

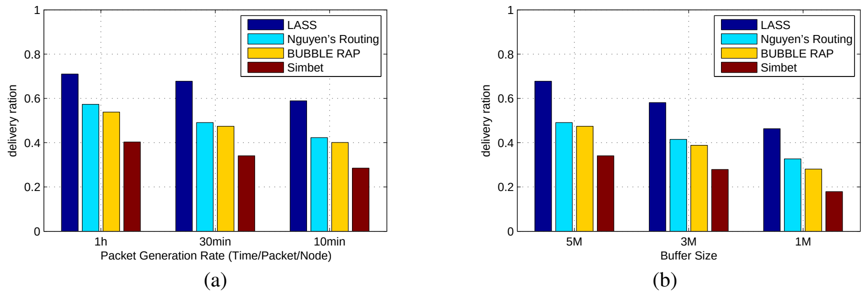

Fig. 11: Comparison results under different parameter settings. Figure(a) differs the packet generation rate. Figure(b) differs the buffer size.

## Appendix I
## Experiments under Different Parameter Settings

In this section, we choose different parameters to test the performance of LASS and other social-based methods. The packet size is 100KB.

First, under 5MB buffer size, we change the packet generation rate from 1h per packet per node to 10min per packet per node. Fig.11(a) shows the result. When the workload increases, all methods have lower packet declivity ratio. However, they change at different rates. When the packet generation rate is 1h per packet per node, the delivery ration of LASS is higher than Nguyen's Routing with 24 percent. When the rate increases to 10min per packet per node, the delivery ration of LASS is higher than Nguyen's Routing with 39 percent. This means that LASS is more efficient under high workloads.

Second, under 30min per packet per node, we change the buffer size from 5MB to 1MB. Fig.11(b) shows the result. When the buffer size decreases, all methods have lower packet declivity ratio. However, they change at different rates. When the buffer size is 5MB, the delivery ration of LASS is higher than Nguyen's Routing with 37 percent. When the buffer size is 1MB, the delivery ration of LASS is higher than Nguyen's Routing with 41 percent. This means that LASS is efficient under small buffer size.

## Appendix J
## Related Work

On the one hand, some studies have shown that exploiting social relationships can achieve better data forwarding performances (our work is belonged to this kind). Daly and Haahr [3] proposed SimBet data forwarding algorithm in delay tolerant MANETs. It uses betweenness centrality and social similarity to increase the probability of a successful data forwarding. Authors show that SimBet performs well, especially when the connectivity is low. However, it does not consider contact frequencies between node pairs. Hui *et al.* [4] proposed an algorithm called BUBBLE RAP in DTNs, with making use of node centrality and weighted k-clique community structure to enhance delivery performance. It is better than Daly and Haahr [3]. But it needs to give a priori value of $k$ to identify meaningful community structure, which is impractical in mobile social networks. Moreover, it has the same problem with [3], *i.e.*, using betweenness to calculate global and local centrality, without considering node encounter probability. Gao *et al.* [5] studied multicast in DTNs from the social network perspective. With known community structures, authors formulates the relay selection as a unified knapsack problem. But this method assumes that community structures are already known and some parameters' optimization requires global information to support. Fan et al. [31] studied a geo-community-based broadcasting scheme for mobile social networks by exploiting node geo-centrality and geo-community. Nguyen *et al.* [7] proposed an overlapping community based data forwarding algorithm, called Nguyen's Routing. An efficient community detection method is designed for tracing the evolution of the overlapping communities in mobile networks. Taking advantage of the overlapping community structure, Nguyen's Routing uses the number of common interests as social similarity to design data forwarding scheme. However, it only focuses on the binary graph and does not consider the node local activity.

On the other hand, from another perspective, some studies have demonstrated the social relationships limit the freedom transmission between two nodes. Li et al. [6] introduced socially selfish properties into data forwarding scheme in delay tolerant networks, where protocol SSAR considered both users' forwarding willingness and their contact opportunity. Li et al. [8] studied a joint rate control, routing, and capacity allocation scheme to achieve optimal multirate multicast in dynamic wireless networks, which addressed social selfishness of users by differentiating relay costs towards different destinations. Lin *et al.* [9] proposed a PrefCast algorithm. It considers users' heterogeneous preferences for different content objects in mobile social dissemination, and meanwhile produces the maximal total utility for all users. Wu *et al.* [10] proposed a social feature-based multipath routing scheme in DTNs. It is based on the idea that the social features will play an important role in data forwarding in social contact networks. Finally, the scheme makes the routing problem become a hypercube-based feature matching process.