Distributed Community Detection in Delay Tolerant Networks

Pan Hui, Eiko Yoneki, Shu-Yan Chan, Jon Crowcroft
University of Cambridge, Computer Laboratory
Cambridge CB3 0FD United Kingdom
[firstname.lastname@cl.cam.ac.uk]

ABSTRACT

Community is an important attribute of Pocket Switched Networks (PSN), because mobile devices are carried by people who tend to belong to communities. We analysed community structure from mobility traces and used for forwarding algorithms [12], which shows significant impact of community. Here, we propose and evaluate three novel distributed community detection approaches with great potential to detect both static and temporal communities. We find that with suitable configuration of the threshold values, the distributed community detection can approximate their corresponding centralised methods up to 90% accuracy.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer Communication Networks—*Distributed Systems*; I.6 [**Computing Methodologies**]: Simulation and Modeling

General Terms

Measurement, Experimentation, Algorithms

Keywords

Distributed Community Detection, Delay Tolerant Networks, Network Measurement, Social Networks

1. INTRODUCTION

Pocket Switched Network (PSN) [1] is a kind of Delay Tolerant Networks (DTN) [6] that provides intermittent communication for humans carrying mobile radio devices. Traditional naive multiple-copy-multiple-hop flooding schemes have been empirically shown to work well in dense environments such as academic conferences, and they provide fair performance in sparse settings, such as city-wide communications, in terms of delivery ratio and delay [1]. However, in terms of delivery cost, the naive approach is far from satisfactory because it creates a lot of unwanted traffic as a side-effect of the delivery scheme, and the overhead rapidly becomes unacceptable in a contentious, vulnerable and resource-scarce mobile network.

In the research community, it is widely believed that identifying community information about recipients can help select suitable forwarders and reduce the delivery cost compared to naive "oblivious" flooding. This is a reasonable intuition, since people in the same community are likely to meet regularly and hence be appropriate forwarders for messages destined for other members of their community. Similarly, in the area of sociology, the idea of "correlated interaction" is that an organism of a given type is more likely to interact with another organism of the same type than with a randomly chosen member of the population [22]. If the correlated interaction concept applies, then our intuition is that using this community information to influence forwarding paths may be advantageous. Hence identifying the local communities of each mobile device can be important to improve forwarding efficiency in PSN.

Many centralised community detection methods have been proposed and examined in the literature (see the recent review papers by Newman [21] and Danon *et al.* [3]). These centralised methods are useful for offline data analysis on mobility traces collected to explore structures in the data and hence design useful forwarding strategies, security measures, and killer applications. But as self-organising networks, we would also ask whether the mobile devices can sense and detect their own local communities instead of relying on a centralised server, which leads to the area of distributed community detection. Clauset [2] defines a measure of local community structure and an algorithm that infers the hierarchy of communities that encloses a given vertex by exploring the graph one vertex at a time. Although its original design was for graphs with a known topology instead of dynamics temporal graphs such as PSN, it provides a motivation for us to examine different centralised community detection algorithms and investigate the possibility of developing a distributed version.

The rest of this paper is structured as follows. We introduce the experimental data sets in Section 2, followed by a summary of community detection methods and routing methods in Section 3. We discuss three distributed community detection algorithms in Section 4. We evaluate the distributed algorithms against centralised algorithm in Section 5, and introduce some similarity measures. Finally we conclude the paper with a brief discussion.

MobiArch'07, August 27–31, 2007, Kyoto, Japan.

2. EXPERIMENTAL DATASETS

The experimental data we used were gathered by the Haggle Project [4], MIT Reality Mining Project [5] and UCSD wireless experiment [19].

- In *Cambridge*, the iMotes were distributed mainly to two groups of students from University of Cambridge Computer Laboratory, specifically undergraduate year-one and year-two students, and also some PhD and Masters students.
- In *Reality*, 100 smart phones were deployed to students and staff at MIT over a period of 9 months. These phones ran software that logged contacts with other Bluetooth enabled devices by performing Bluetooth device discovery every five minutes, as well as logging information about the cellular tower with which they are associated (a total of 31,545 different towers were logged).
- In *UCSD*, PDAs were used to log the visibility of access points (APs) from WiFi networks, for a duration of three months. We required data about device-to-device transmission opportunities, so the raw data sets were pre-processed as in [1].

Previously the characteristics of these data, such as inter-contact and contact distribution, have been explored in several studies [1] [16], to which we refer the reader for further background information. Table 1 summarises these three datasets:

Experimental data set	Cambridge	UCSD	Reality
Device	iMote	PDA	Phone
Network type	Bluetooth	WiFi	Bluetooth
Duration (days)	11	77	246
Granularity (seconds)	600	120	300
Number of Experimental Devices	54	273	97
Number of External Devices	11,357	NA	NA
Number of internal contacts	10,873	195,364	54,667
Average # Contacts/pair/day	0.345	0.034	0.024
Number of external contacts	30,714	NA	NA

Table 1: Characteristics of the three experiments

3. RELATED WORKS

Community detection in complex networks has attracted a lot of attention in recent years. Community structures are usually substructures/subgraphs corresponding to important functions, and examples can be found in many areas, such as World Wide Web [7], biological networks [9], social networks [21], and also the Internet [18]. In PSN, community structure would correspond to human communities or some structures which are beneficial for forwarding efficiency [12]. The recent reviews [21] and [3] may serve as introductory reading in community detection methods. Besides the methods mentioned in the two reviews papers, we also introduce the k-CLIQUE community detection method by Palla *et al.* [23] and the weighted community analysis methods by Newman [13], which are used to compare with our distributed algorithms in this paper.

For routing and forwarding in DTN and mobile ad hoc networks, there is much existing literature. Vahdat *et al.* proposed the epidemic routing [26] which is similar to the "oblivious" flooding scheme we evaluated in this paper. Spray and Wait [25] is another "oblivious" flooding scheme but with a self-limited number of copies. Grossglauser *et al.* proposed the two-hop relay schemes [8] to improve the capacity of dense ad hoc networks. Lindgren *et al.* proposed PROPHET [17], which is a probability routing scheme based on the very early belief that community will help with routing decisions. There are also many other varied schemes such as the adaptive routing [20] by Musolesi *et al.*, the practical routing scheme [14] by Jones *et al.* and Mobyspace [15] by Leguay *et al.*, these are all examples of how to use system and mobility information to improve the efficiency of routing and forwarding from "oblivious" flooding. So far, there are few empirical evaluations of the impact of community information on forwarding efficiency except a very early study by Hui *et al.* [12] based on a *priori* affiliation information and a technical report [11] based on centralised detected communities.

4. DISTRIBUTED COMMUNITY DETECTION

Human society is composed of many relationships and communities. Two persons can be strangers, familiar strangers, community members, friends or families. We think contact duration and number of contacts, which are correlated to familiarity and regularity, are two key criteria by which to categorize these relationships.

Figure 1 shows these four kinds of relationship from the Cambridge data. This categorisation may not be very accurate but it did give us some hints on how to use our mobile devices to infer different categories of social contexts from our daily contact patterns by setting thresholds for the contact duration and number of contacts.

In this section, we will introduce three distributed community detection algorithms, named SIMPLE, k-CLIQUE, and MODULARITY. We will first introduce definitions and terms, followed by the details of three algorithms we propose here.

4.1 Definitions

The common terminologies for all these algorithms are:

Familiar set: we assume each vertex (mobile device) will keep a map of vertices they have encountered with the cor-

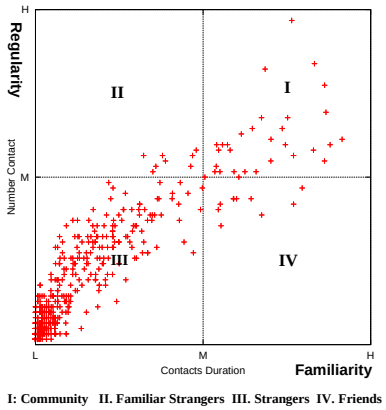

Figure 1: Four categories of relationship (Cambridge)

responding cumulative contact durations. When the cumulative contact duration with a vertex exceeds a certain threshold T_{th}, it is promoted to be included into its *familiar set* F. In the language of graph theory, these two vertices now have an undirected edge between them. For a given vertex, v_i, it has perfect knowledge of its own familiar set (by definition), denoted F_i. The same vertex also could have gathered some incomplete knowledge of other vertices' familiar sets, e.g. a local approximation of the familiar set for vertex v_j would be denoted $\tilde{F}_j$.

Local Community: a vertex's local community, denoted by *C*, contains all the vertices in its *familiar set* (its direct neighbors) and also the vertices that are selected by our following community detection algorithms (the selection criteria of each algorithm to be further elaborated). Because of temporal non-synchronisation, each vertex supposed to be in the same community may have detected a different local community.

The basic structure of our algorithms is as follows. When a mobile device v_0 first initialises its community detection procedure, the local community C_0 only contains this source vertex. Whenever it encounters another device v_i, they will exchange part of their local knowledge of the network. v_0 then has to decide on the following based on certain acceptance criteria:

1. whether to place the encountered vertex v_i in its *familiar set* F_0 and/or *local community* C_0.
2. whether C_0 should merge with the whole or part of C_i.

All the three algorithms we introduce here differ only in the admission criteria into the familiar set, local community and merging of communities above.

Before we move to next section, we need to give a brief introduction to MODULARITY. MODULARITY is a variation of Clauset's community detection using local modularity [2]. In the paper, Clauset defines a measure of local community structure and an algorithm that infers the hierarchy of the communities that enclose a given vertex by exploring the graph one vertex at a time. In this case, we need to treat the PSN as unexplored relation graph. The following terminology is relevant to MODULARITY:

Adjacency set: the *adjacency set* of a particular local community C_0 is denoted u_0. In terms of graph theory, it is the set of vertices which are outside the local community C_0, but each vertex in it is adjacent (has direct edges connecting) to one or more members of the local community C_0, e.g.

$$u_0 = \{v_i \mid v_i \in \bigcup_{v_j \in C_0} F_j \setminus C_0\}$$

Boundary Set: in terms of graph theory, for a given vertex v_0 and its local community C_0, the associated boundary set B_0 is defined as the subset of vertices in C_0, whose members have edges connecting to one or more vertices outside C_0, i.e.

$$B_0 = \{v_i \mid (v_i \in C_0) \wedge ((F_i \setminus C_0) \neq \emptyset)\}$$

Boundary-Adjacency Matrix: for a vertex v_0 and its boundary set B_0, the associated boundary-adjacency matrix of $\mathcal{B}_0$ is defined as follows: $(\mathcal{B}_0)_{ij} = 1$ if vertex v_0 has knowledge that the vertices v_i and v_j are connected, and at least one of them is in the boundary set B_0, otherwise $(\mathcal{B}_0)_{ij} = 0$; i.e. $(\mathcal{B}_0)_{ij} = 1$ *iff*

$$(v_i \in B_0 \vee v_j \in B_0) \wedge (v_i \in \tilde{F}_j \vee v_j \in \tilde{F}_i)$$

Local Modularity: for a given vertex v_0, the entire local knowledge, $\tilde{G}_0$, it has of the network, comprises the vertices in C_0 and u_0, together with its partial knowledge of the connections between those vertices. Clauset introduced Local Modularity, so that each vertex can measure the sharpness of its local community boundary, and the measures are independent of the size of the enclosed communities. The local modularity R for a given C_0 with B_0 is defined

$$R_0 = \frac{\sum_{s,t}(\mathcal{B}_0)_{st}}{\sum_{i,j}(\mathcal{B}_0)_{ij}} = \frac{I}{|T|}, \quad (1)$$

where $\forall s,t\ v_s, v_t \in C_0$, $\forall i,j\ v_i, v_j \in C_0 \cup u_0$, and hence *T* is the set of edges in $\tilde{G}_0$ with one or more endpoints in B_0, while *I* is the number of those edges with neither endpoints in u_0. If $B_0 = \emptyset$, R_0 is defined to have value of 1.

Δ Local Modularity: when adding a new vertex $v_j \in u_0$ to an existing community C_0 with B_0, the change in R value, ΔR_0, can be calculated by

$$\Delta R_0 = \frac{x - R_0 * y - z(1 - R_0)}{|T| - z + y}, \quad (2)$$

where *x* is the number of edges in *T* that terminate at v_j, *y* is the number of edges that will be added to *T* by the agglomeration of v_j and *z* is the number of edges that will be removed from *T* by the agglomeration.

4.2 Algorithms

In the following algorithms, we will use SIMPLE, k-CLIQUE, and MODULARITY to denote the variations of the skeleton algorithm. The main differences are that k-CLIQUE and MODULARITY have more data to maintain and there are some changes to step 5 and step 6. When a mobile device v_0 encounters another device v_i, the following algorithm will execute:

1. Each vertex, v_0, needs to maintain the following information: a list of encountered nodes and their contact durations (practically encounters that do not meet certain criteria will be discarded from the list), its *familiar set* F_0 (its familiar set of vertices), its local community C_0 detected so far, and

   (k-**CLIQUE**) a local approximation of the *F*amiliar *S*ets *o*f all vertices in its *L*ocal *C*ommunity C_0:

   $$FSoLC_0 = \{\tilde{F}_j \mid v_j \in C_0\}$$

   (**MODULARITY**) as in k-CLIQUE: $FSoLC_0$.

2. *Initialisation:* C_0 is set to $\{v_0\}$, $F_0 = \emptyset$ and $FSoLC_0 = \emptyset$

3. When v_0 encounters another v_i, they exchange local information, i.e. v_0 will acquire from v_i the following: C_i, F_i and

   (k-**CLIQUE**) $FSoLC_i$

   (**MODULARITY**) $FSoLC_i$. v_0's first use of this newly acquired information is to improve its own approximation of $\tilde{F}_i$ and $FSoLC_0$ Each local approximation of familiar set in $FSoLC_0$ is merged (by taking the set union) with the corresponding version in $FSoLC_i$ just obtained from v_i.

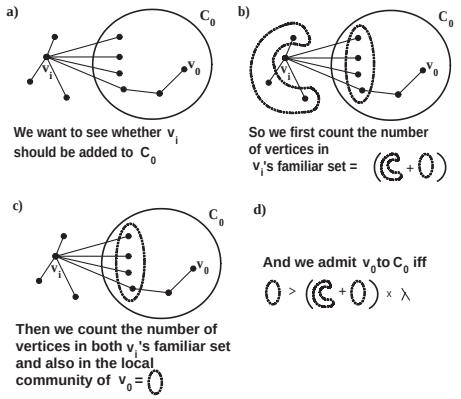

Figure 2: Admission criteria for SIMPLE

4. If v_i is not in F_0, v_0 updates the total contact duration counter of v_i which is stored at v_0, until v_i falls out of contact and meanwhile the algorithm forks and proceeds to the next step (5). When the total contact duration count has exceed a certain threshold (a design parameter), v_0 will insert v_i in F_0 and C_0, (**MODULARITY**): also add F_i to (or merge with existing entry in) $FSoLC_0$, then the algorithm proceeds to step 6.

5. If v_i is not in C_0, then add v_i to C_0 if it satisfies the following algorithm-specific criteria:

   (**SIMPLE**) if $|F_i \cap C_0|/|F_i| > \lambda$ (where λ is the merging threshold which we will vary in this paper to see the different of final communities detected). See Figure 2.

   (k-**CLIQUE**) if the *familiar set*, F_i contains at least $k-1$ members of the local community, C_0, (see Figure 3), i.e.

   $$|F_i \cap C_0| \geq k - 1.$$

   (**MODULARITY**) if $(F_i \neq \emptyset) \wedge ((F_i \subseteq C_0 \wedge B_0 \neq \emptyset) \vee \Delta R_0 > 0)$(the difference between the local modularity measure before and after adding v_i to C_0 is *+ve*, see Figure 4).

6. If v_i is added to C_0 in the previous steps, the aggressive variants of the algorithm behave as follows:

   (**SIMPLE**) if the number of vertices overlapping C_0 and C_i, (i.e. $|C_0 \cap C_i|$), is higher than γ of $|C_0 \cup C_i|$ (γ is the merging threshold as well which can be different from λ in step 5, but we will use the same value for both cases in this section), then merge (by taking the set union of) the two communities. i.e. the merging criteria is

   $$|C_0 \cap C_i| > \gamma |C_0 \cup C_i|$$

   (k-**CLIQUE**) consider each vertex v_j inside C_i (the local community of v_i), if its *familiar set*, $\tilde{F}_j$ contains at least $k-1$ members of C_0 (the local community of v_0), v_j is added to the local community C_0, i.e. if

   $$|\tilde{F}_j \cap C_0| \geq k - 1$$

   If this criteria is satisfied, then $FSoLC_0$ also needs to be updated to include $\tilde{F}_j$

   (**MODULARITY**) the algorithm only considers adding the vertices in the set K :

   $$\{v_k \mid \exists j \; s.t. \; v_j \in C_0 \cap C_i \wedge v_k \in \tilde{F}_j \wedge v_k \in C_i \setminus C_0\}$$

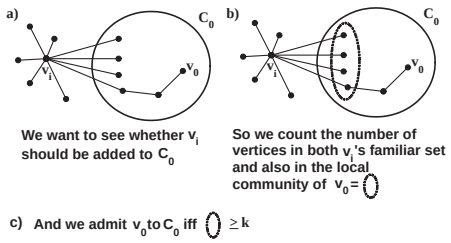

Figure 3: Admission criteria for k-CLIQUE

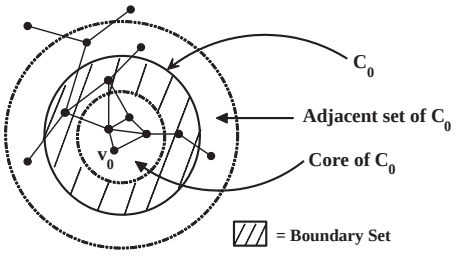

Figure 4: The higher the local modularity of a community, the fewer the number of edges connecting it to its Adjacent Set. A community has a Local Modularity value of 1 when it has an empty Boundary Set.

Figure 5 illustrates how and why set K is chosen. The shaded area in Figure 5(a) shows the vertices in the set K. When considering merging parts of two communities together, one first considers locating all the vertices that are common to the local communities of both vertices ($C_0 \cap C_i$, shaded area in Figure 5(b)). Then we consider those vertices that are adjacent to the set of vertices in $C_0 \cap C_i$ (shaded area in Figure 5(c)), which are more closely connected to the common vertices of the two local communities and hence suitable for merging into C_0. But since portion of them are already in C_0, and portion of them are in neither of the local communities, so we only consider the set of vertices in K. During the encounter with v_i, v_0 acquired the information $\tilde{F}_k$ for all vertices in K from v_i, and now for each $v_k \in K$,

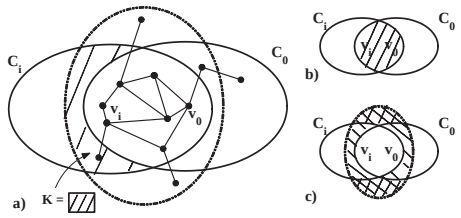

Figure 5: Explanation of set K in MODULARITY in step 6

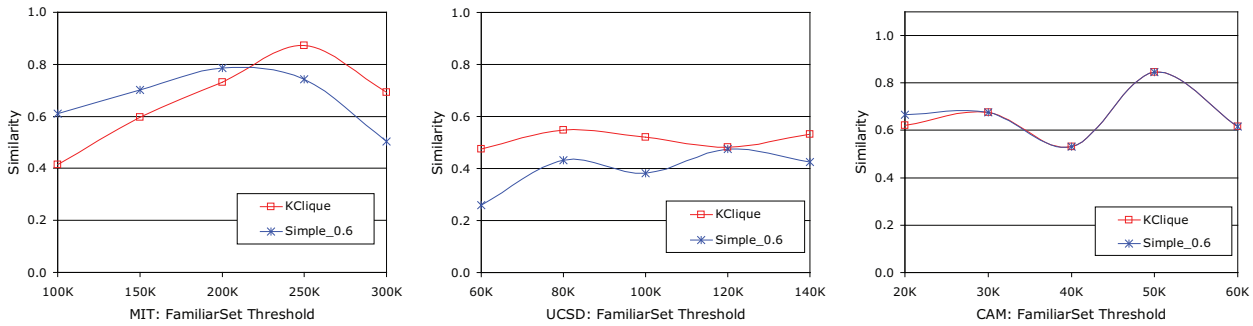

Figure 6: Impact of FamiliarSet threshold, k-CLIQUE and SIMPLE

it evaluates whether $\tilde{F}_k \subseteq C_0$. If this condition is satisfied, the corresponding v_k is added to C_0. For the rest of the v_k, these are then considered in descending order of ΔR_0 (the difference in the R value before and after adding v_k to C_0). Vertices with a negative or zero contribution to ΔR will not be added to C_0, and the values of ΔR are re-evaluated after each addition to C_0. For each addition of v_k to C_0, $\tilde{F}_k$ is also added to the set $FSoLC_0$.

Clearly, the SIMPLE algorithm requires less storage and less computation. The k-CLIQUE algorithm is in the middle and MODULARITY is the most demanding one - because of the need to re-evaluate ΔR in each iteration, hence in Step 6 of the algorithm only part of the community (K) is considered to be merged, as a resource/performance tradeoff.

5. RESULTS AND EVALUATIONS

In this section, we evaluate the communities detected by the distributed methods against the centralised methods. In order to do the comparison, we need to first develop the similarity measurements.

5.1 Similarity Measures

Newman [21] introduce a metric called *fraction of vertices correctly identified* to evaluate the communities detected against the pre-known communities. Another measurement metric is used in [3] by Danon *et al.*, which is called *normalised mutual information*(NMI) measure.

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} N_{ij} log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\sum_{i=1}^{c_A} N_{i.} log(\frac{N_{i.}}{N}) + \sum_{j=1}^{c_B} N_{.j} log(\frac{N_{.j}}{N})} \quad (3)$$

where c_A is the number of real communities, c_B is the the number of found communities, $N_{i.}$ is the sum over row *i* of matrix N_{ij} and $N_{.j}$ is the sum over column *j*.

We can adapt these two measurements to evaluate our distributed community detection algorithm against its corresponding centralized method. However before moving forward, we need to first consider the fairness problem about comparison. Since, for distributed community detection, each node will detect the local communities to which it belongs to, if a system has N nodes, there would be at least N communities detected (i.e. $c_B \geq N$). Denote the number of real communities as c_A as above, if $N \gg c_A$, the evaluation of c_A against c_B would be unfair, especially if over weighted by the big community. Also, considering that the network is a temporal graph and some nodes are more popular than others, nodes belonging to the real community may not have the same local view of the communities detected. Therefore we need to consider a modification to address this problem. Our approach is to chose the biggest detected community, move it to the *core community* list and then discard the communities detected by all the nodes included in it, and then repeat this for the remaining biggest one on the list and continue until no more communities are left. We then evaluate c_A against the *core community* list. The biggest communities are not necessarily the best communities detected: they may contains a lot of redundance so our selection of the biggest communities does not necessarily favor our algorithms. This shrinking process will remove the smaller groups of the overlapping communities, which may also penalise our results.

Considering Newman's method is little bit harsh as he mentioned in his paper there are cases in which one might consider some of the vertices to have been identified correctly, and this method would not. Also considering that for all three data sets, there are many more single node communities than bigger communities, this will make the NMI measure tend towards 1. Hence here we consider another modified similarity measurement. Here we introduce the similarity by using the classic Jaccard index [24] which was proposed by Jaccard over hundred years ago to evaluate the similarity of two communities.

$$\sigma_{Jaccard} = \frac{|\Gamma_i \bigcap \Gamma_j|}{|\Gamma_i \bigcup \Gamma_j|} \quad (4)$$

where Γ_i is the members of community *i* and $|\Gamma_i|$ is the cardinality of the set Γ_i, that is equal to the number of member in community *i*. In this paper, we will compare the *core communities* detected by distributed methods with the communities detected by centralised algorithms using this similarity measurement.

5.2 Results of detection

To evaluate the community detection algorithm, we replay the mobility traces of the three experiments and emulate the gossiping of community information on each encounter as the algorithms described in section 4.2. Here we only evaluate the communities detected after the whole traces, which lasted 9 months for *Reality*, 3 months for *UCSD* and 11 days for *Cambridge*, are replayed. As a first step we do not evaluate the time need for the communities to be well developed at the middle of the emulation.

Figure 6 shows the similarity between the communities detected by distributed SIMPLE method and k-CLIQUE against

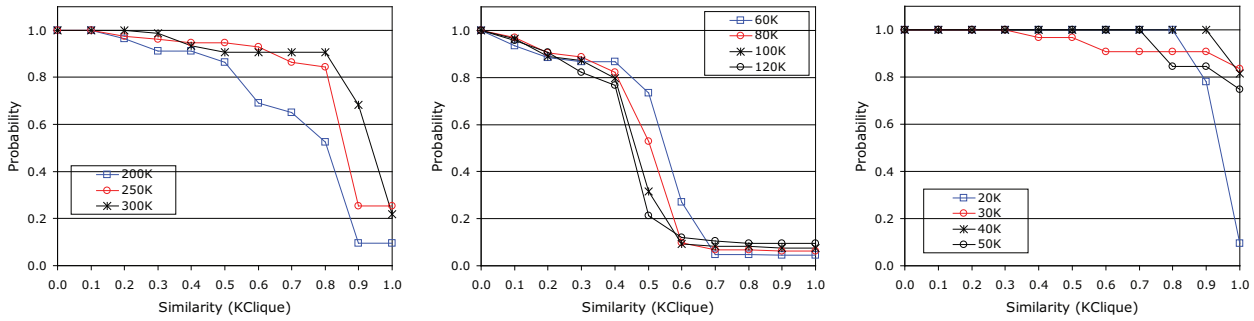

Figure 8: Distribution of local community views

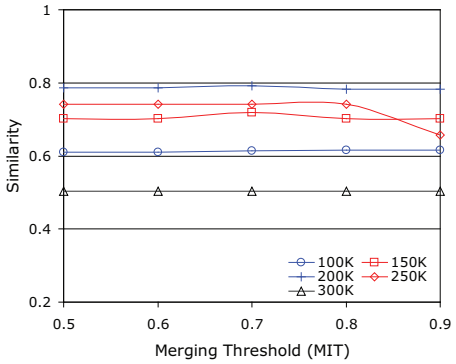

Figure 7: Impact of merging threshold

the communities detected by the centralised k-CLIQUE algorithm with a threshold of 389k seconds for *Reality* dataset, 78k seconds for *UCSD*, and 36k seconds for *Cambridge*. These threshold values for the centralised methods were selected following many trials and also by studying the nature of the group of experimental objects. Some of them are found to agree with the real experimental groups, such as for the *Cambridge* data, the two groups detected correspond to the two main participant groups. Figure 6 also shows the different similarity values with different familiar set thresholds. We can see that the k-CLIQUE methods shows better results most of time than the SIMPLE method. With a suitable threshold, the distributed algorithms for both SIMPLE and k-CLIQUE can reach around 80% of the performance of the centralised algorithm. For the SIMPLE case, we use a merging threshold, λ, of 0.6. We also find out that varying the merging threshold from 0.5 to 0.9 makes little difference; whereas the Familiar Set threshold changes the similarity values quite significantly. Figure 7 shows an example for the *Reality* data using the SIMPLE approach.

Since we know the network is highly intermittently connected, the local community information for each node within the same community may not be synchronised. We want to know how different these local community views are. From the *Core Communities*, we compare the local community detected by each member in each *Core Community* with its *Core Community*, we calculate the similarity values and then plot the distributions of all these similarity values. Figure 8 shows these distributions for the three datasets (from left to right, they are *Reality*, *UCSD* and *Cambridge* respectively) using distributed k-CLIQUE. We can see that for both *Reality* and *Cambridge*, the local community views are quite similar to the selected largest *Core Community*, this would be probably because of relatively smaller dataset size and higher connectivity. And these two groups of nodes also agree with the two main groups of students participating the experiment.The *UCSD* data relies on users connecting to the centralised access points instead of peer contacts and hence is more sparse. In general, the local community views have bigger variation when using SIMPLE. Figure 9 gives an example of the *Reality* case, similar variations are also observed in the other two datasets. And we also find out that there is no impact of the merging threshold, λ (from 0.5 to 0.9) on the distribution if using the same Familiar Set threshold.

Figure 10 also shows the comparisons of the MODULARITY and SIMPLE with the the centralised Newman weighted network analysis [13]. Since the MODULARITY and the Newman method both used modularity, it is fairer to compare them than comparing with the centralised k-CLIQUE method. We can see that in the *Reality* case, MODULARITY has better performance than SIMPLE with the same threshold, and it has a best performance of around 80% similarity. The SIMPLE approach also generally has good performance with a 75% similarity for a suitable threshold setting. For *UCSD*, the similarity values are in general low for both algorithms and this is also true for the k-CLIQUE algorithm, this is probably because of the characteristics of the dataset. For the *Cambridge* data, at high threshold values, MODULARITY behaves better than SIMPLE and the other way round at low threshold values, but they both reach at a maximum point of above 80%.

We conclude this section with Table 2 which summarizes the highest similarity values calculated by each distributed algorithm. For SIMPLE, we show both its comparison with

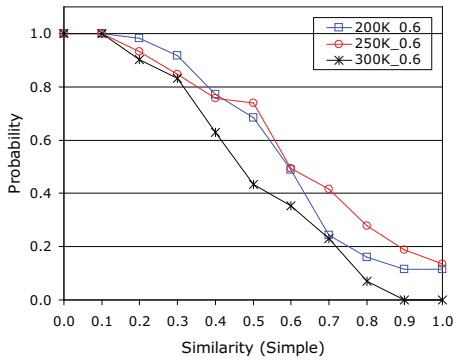

Figure 9: Distribution of local community views SIMPLE

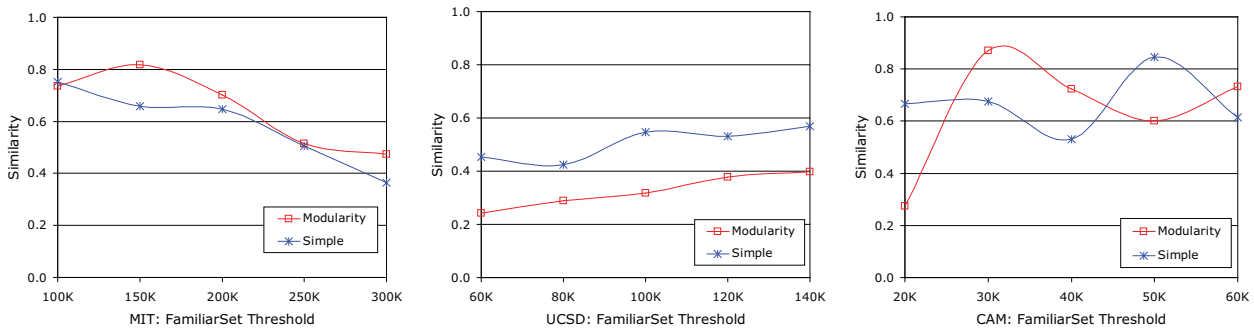

Figure 10: Impact of FamiliarSet threshold, MODULARITY and SIMPLE

the centralised k-CLIQUE (first) and the centralised Newman method (second). We can see that generally MODULARITY and k-CLIQUE have slightly better performance than their SIMPLE counterpart. That is to be expected since they require more information and calculation, especially the computation complexity of MODULARITY is $O(n^4)$ in the worst case, where n is the size of the network explored so far. However, since a factor of n^2 is contributed by the evaluation of each ΔR, which in reality is likely to be bounded by $O(k^2)$ where k is the average degree of a vertex in the graph, the worst case performance is thus $O(n^2k^2)$. Considering its computational and storage requirements, the performance of SIMPLE is quite acceptable, so we would suggest SIMPLE, with $O(n)$, for the mobile devices with strong constraints on storage and computational complexity. If the mobile devices can afford the storage for a local copy of the Familiar Set of its community members, k-CLIQUE would be a good choice for its reasonably good similarity values and also quite low computational complexity, $O(n^2)$ in the worst case. MODULARITY requires the most computational power but it has no significant better performance in these cases. This maybe biased by the limitations of the experimental datasets, but we will not strongly recommend it at this moment.

Experimental data set	SIMPLE	k-CLIQUE	MODULARITY
Reality	0.79/0.76	0.87	0.82
UCSD	0.47/0.56	0.55	0.40
Cambridge	0.85/0.85	0.85	0.87

Table 2: Summary of best performance of the algorithms

5.3 Limitations

There are several limitations to our study in this paper, and we want to point them out here:

- As a first study, we only evaluate the communities detected after the replaying of the whole traces, but did not evaluate the communities at different stages of the emulation. Evolution of the communities detected at different times could also be an interesting study topic.
- The Familiar Set threshold values we used in the emulations are traces dependent and were chosen based on the whole duration of a trace. In a real application, we may want to specify them in more general terms such as number of hours or number of times per day, per week or per month. Here we just want to compare the performance of the distributed algorithms with the centralised ones so we simply specify them in relative to the whole experimental durations.
- We did not evaluate the detection of different categories of relationship in this paper. In real applications, however, the mobile devices should be able to detect the different categories of relationship in Figure 1 by specifying the Familiar Set thresholds for contact durations and number of contacts.
- In the current version of the algorithms, we need to specify a static Familiar Set threshold, but maybe in future versions some more dynamic methods such as fuzzy logic or other AI methods could be used to reduce manual configuration.
- We didn't consider *aging* of the contacts at this moment, but we need to look into it in the future. Some previous contacts may become irrelevant after some time, but which takes up storage and may cause false-positive impact for the detection, so good *aging* mechanisms about the contacts should be considered.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed three distributed community detection algorithms with different levels of computational complexity and resource requirements. We evaluated them on three human mobility experimental datasets and discovered that the communities detected by the distributed algorithms can satisfactorily approximate the centralised algorithms which require the whole network topology. We compared the performance of these three algorithms and proposed a scenario in which each could be used.

In this future, we would like to evaluate our algorithms on more mobility traces such as the WiFi traces from the Crawdad project, and also some forthcoming iMote experiments to make more conclusive statement about the accuracy and application scenarios of these algorithms. And we would also like to develop our studies further with regard to the limitations we listed in section 5.3.

Detail analysis of the impact of the detected communities on the PSN forwarding efficiency can be found in our technical report [11].

7. ACKNOWLEDGEMENT

This work was supported in part by the Haggle Project under the EU grant IST-4-027918. We would like to acknowledge comments from Derek Murray. We would also like to acknowledge CRAWDAD project [10] for their hosting and sharing of the mobility data.

8. REFERENCES

[1] A. Chaintreau et al. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. INFOCOM*, April 2006.

[2] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72:026132, 2005.

[3] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification, 2005.

[4] C. Diot et al. Haggle project,http://www.haggleproject.org, 2004.

[5] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, V10(4):255–268, May 2006.

[6] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. SIGCOMM*, 2003.

[7] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[8] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *Transactions on Networking*, 10(4):477–486, August 2002.

[9] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl), December 1999.

[10] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proc. Mobicom*, 2004.

[11] P. Hui and J. Crowcroft. Bubble rap: Forwarding in small world dtns in ever decreasing circles. Technical Report UCAM-CL-TR-684, University of Cambridge, May 2007.

[12] P. Hui and J. Crowcroft. How small lables create big improvements. In *Proc. IEEE ICMAN*, March 2007.

[13] M. E. J.Newman. Analysis of weighted networks. *Physical Review E*, 70:056131, 2004.

[14] E. P. C. Jones, L. Li, and P. A. S. Ward. Practical routing in delay-tolerant networks. In *Proc. WDTN*, 2005.

[15] J. Leguay et al. Evaluating mobility pattern space routing for DTNs. In *Proc. INFOCOM*, 2006.

[16] J. Leguay et al. Opportunistic content distribution in an urban setting. In *ACM CHANTS*, 2006.

[17] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proc. SAPIR*, 2004.

[18] D. Lusseau and M. E. J. Newman. Identifying the role that individual animals play in their social network. *PROC.R.SOC.LONDON B*, 271:S477, 2004.

[19] M. Mcnett and G. M. Voelker. Access and mobility of wireless pda users. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):40–55, April 2005.

[20] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Proc. WOWMOM*, 2005.

[21] M. Newman. Detecting community structure in networks. *Eur. Phys. J. B*, 38:321–330, 2004.

[22] S. Okasha. Altruism, group selection and correlated interaction. *British Journal for the Philosophy of Science*, 56(4):703–725, December 2005.

[23] G. Palla et al. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[24] P.Jaccard. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547, 1901.

[25] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile n etworks. In *Proc. WDTN*, 2005.

[26] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.