Transient Community Detection and Its Application to Data Forwarding in Delay Tolerant Networks

Xiaomei Zhang and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA, 16802
Email: {xqz5057,gcao}@cse.psu.edu

***Abstract*—Community has received considerable attention because of its application to many practical problems in mobile networks. However, when considering temporal information associated with community (i.e., transient community), most existing community detection methods fail due to their aggregation of the contact information into a single weighted or unweighted network. In this paper, we propose a *contact-burst-based* clustering method to detect transient communities by exploiting the pairwise contact processes. In this method, we formulate each pairwise contact process as regular appearance of contact bursts, during which most contacts between the pair of nodes happen. Based on such formulation, we detect transient communities by clustering the pairs of nodes with similar contact bursts together. We also propose a new data forwarding strategy for delay tolerant networks in which transient communities serve as the data forwarding unit. Evaluation results show that our strategy can achieve much higher data delivery ratio than traditional community-based strategies with comparable network overhead.**

I. Introduction

In Delay Tolerant Networks (DTN) [1], mobile devices are only intermittently connected due to mobility and low node density. As a result, it's hard to maintain an end-to-end path, which makes data forwarding in DTN extremely difficult. To address this problem, researchers have proposed approaches to exploit social network concepts such as centrality [2][3][4], community [5][6][7][8] and friendship [9][10][11].

Community has received considerable attention because of its applications to data forwarding in DTN, worm containment [12], etc. However, it is also a challenge to detect communities in a large network [13] [14], especially considering the temporal information associated with community. For example, a class community consisting of students attending a class may only appear during daytime, and a dormitory community consisting of students at a dormitory may only appear at night. Since these communities normally appear during a time period and disappear thereafter, they are referred to as *transient communities (TCs)*. Although many community detection methods have been proposed in the literature, there are hardly any methods for TC detection.

Existing community detection methods are generally based on weighted networks or unweighted networks. For example, algorithms have been proposed in [15][16] to detect communities in weighted networks. Methods like label propagation [17] have been proposed to detect communities in unweighted networks. Moreover, Clique Percolation Method (CPM) or called K-clique [18] has been proposed to detect communities in both weighted and unweighted network. AFOCS [7] can detect static communities and track community dynamics based on unweighted network snapshots. However, by aggregating contact information into a weighted or unweighted network, important contact information such as the time when nodes contact will be lost. Losing such temporal information may result in two problems related to TC detection: false mixture and false separation as shown in Figure 1 and Figure 2.

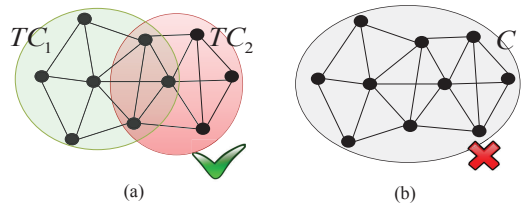


Fig. 1. False mixture: (a) TC_1 appears during daytime and TC_2 appears at night. (b) The two TCs are falsely mixed into one community using a traditional method.

- **False Mixture** There are originally two TCs in the network, as shown in Figure 1 (a). TC_1 is a class community happening during daytime, and TC_2 is a dormitory community happening during night time. The two communities share several students, who take classes together and also live together. Since there is a large overlap between them, traditional community detection methods may falsely mix these two communities as one. For example, CPM (K-clique) and AFOCS fail to distinguish the two communities when the overlap is larger than some threshold.
- **False Separation** Figure 2 (a) shows one TC in the network. Because the network is not strongly connected (i.e., only one node connects the two parts), a traditional method may separate them into two communities. However, they should not be separated since they may indeed attend the same class at the same time.

With false mixture, two highly-overlapping TCs cannot be distinguished by ignoring the temporal contact information.

This work was supported in part by Network Science CTA under grant W911NF-09-2-0053.

978-1-4799-1270-4/13/$31.00 © 2013 IEEE

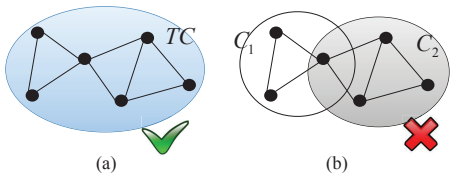


Fig. 2. False separation: (a) The network has one TC. (b) The TC is falsely separated into two communities using a traditional method.

With false separation, one large TC may be falsely separated if node connections are not dense enough. Therefore, traditional community detection methods fail to detect TCs.

In this paper, we propose a *Contact-burst-based Clustering Method (CCM)* to detect TCs by exploiting the pairwise contact processes. It is based on the detailed pairwise contact information between nodes, instead of networks with unweighted on-off edges or weighted edges. In CCM, we formulate each pairwise contact process as regular appearance of *contact bursts*, during which most contacts between the pair of nodes appear. Based on such formulation, we detect TCs by clustering the pairs of nodes with similar contact bursts together. In addition to TC detection, we also apply TCs to data forwarding in DTNs. The contributions of this paper are three-fold:

1) We propose a CCM method to detect TCs. Compared with existing methods such as CPM and AFOCS which do not consider communities' temporal information, our method has much less false mixtures and false separations.
2) A TC may periodically appear, and hence we propose techniques to identify this appearance pattern, which is useful in many applications.
3) We propose a data forwarding strategy in DTNs based on TCs, where data is forwarded to TCs with better relaying capability to the destination considering the time constraint of the data. Evaluation results show that our approach outperforms other existing data forwarding approaches in DTNs.

The rest of the paper is organized as follows. Section II presents our CCM method for TC detection. In Section III, we present the TC-based data forwarding strategy. Section IV gives a brief overview of the related work, and Section V concludes the paper.

II. Transient Community Detection

In this section, we first give some preliminaries, and then present our CCM method for TC detection. Finally, we compare our TC detection method with other community detection methods and propose techniques to identify the periodic appearance patterns of TCs.

A. Preliminary

In this section, we describe the contact traces and introduce some terms that will be used in this paper.

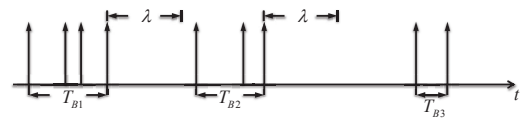


Fig. 3. Three contact bursts. An arrow represents a contact between two nodes.

1) Contact trace: We study the social contact patterns on three sets of traces, Dartmouth campus trace [19], MIT reality trace [20] and UCSD campus trace [21]. These traces record contacts among users carrying mobile devices on campus. In Dartmouth and UCSD traces, devices are WiFi-enabled. The Dartmouth trace uses SNMP logs from Access Points (APs). The original trace contains records over several thousands of WiFi-enabled devices and lasts almost 4 years. In this paper, we focus on the data collected from September 2004 to December 2004. A contact is recorded when two devices detect the same AP simultaneously. The UCSD trace records the WiFi association of human-carried PDA with APs, and a contact is recorded when two devices detect the same AP. In MIT Reality trace, the devices periodically detect their peers via Bluetooth interfaces, and a contact is recorded when two devices move into the communication range of each other. The details of these three traces are shown in Table I.

TABLE I
Trace summary

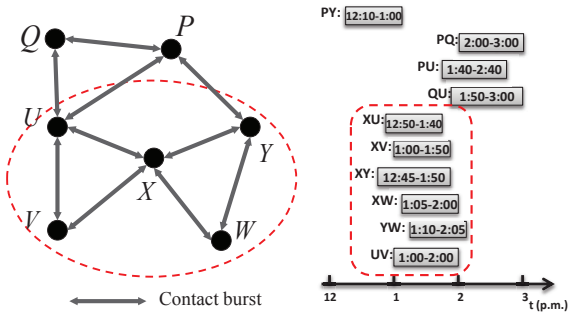
Trace	**Dartmouth**	**MIT Reality**	**UCSD**
Network type	WiFi	Bluetooth	WiFi
Number of devices	425	97	275
Number of contacts	394254	114046	201923
Durations(days)	80	246	78
Granularity(secs)	300	120	20

2) Contact Burst: Simply extracting communities from weighted or unweighted network is not enough to detect TCs. Therefore, instead of simplifying each pairwise contact process to a weight, our method processes the pairwise contact information directly.

From the trace summary, we can see that each trace has hundreds of thousands of contacts and detecting TCs from these contacts directly will be hard. The time complexity will be extremely large, and the opportunistic nature of the contacts hardly provides any clue on how they are related with TCs. Thus, we propose a different solution. We model the pairwise contact process using simple units which can represent the contact information between nodes and can be directly processed. We formulate each pairwise contact process as a series of contact bursts during which most contacts between the pair of nodes appear. A contact burst is defined as follows:

Definition 1: A ***contact burst*** $T_B = [t_s, t_e]$ between two nodes is a time period when contacts frequently appear between these two nodes. Two adjacent contacts belong to one contact burst if and only if the inter-contact time between them is shorter than λ, where λ is a pre-defined threshold.

Figure 3 shows three contact bursts of two nodes, where



(a) (b)

Fig. 4. A connected graph built on contact bursts. Six contact bursts have similar contact durations, and they may be from one TC, circled by red line. The other four have different contact durations, which are not in this TC.

each vertical arrow indicates a contact and T_{B_i} denotes the $i-th$ contact burst. A single contact is not considered as a contact burst because it is more like a random contact.

With the concept of contact burst, each pairwise contact process between two nodes is modeled as a series of contact bursts $T_{B_i} = [t_{s_i}, t_{e_i}], i = 1, 2, 3....$ To verify if the contact bursts can really represent node contacts, we have done some experiments based on the three traces to evaluate how many contacts can be represented by contact bursts. Table II shows the percentage of contacts that are within contact bursts and the length of the contact bursts as a percentage of the total trace duration. The value of λ is set empirically based on different traces. We have found that in all of our three traces, with $\lambda = 1$ hour, most contacts happen within some contact bursts which only account for a small portion of the total time. We have also tested other values of λ, and the results show that the performance is not very sensitive to the change of λ. For simplicity, we set $\lambda = 1$ hour.

TABLE II
HOW CONTACT BURSTS CAN REPRESENT CONTACT PROCESSES ($\lambda = 1$)

Trace	Contacts in bursts (%)	Bursts' length (%)
Dartmouth	66.52%	6.65%
MIT	77.88%	10.36%
UCSD	81.22%	4.93%

Contact bursts are used to find TCs. A contact burst $[t_s, t_e]$ between a pair of nodes is a time interval in which the two nodes frequently contact. In a TC with duration $[t_s, t_e]$, its members usually have frequent contacts with each other within this interval; i.e., their contact bursts are all with duration similar to $[t_s, t_e]$. Therefore, if we are able to find a group of contact bursts with similar duration, and they can mutually form a connected graph and form a TC. For example, in Figure 4, there are 10 contact bursts, where a double arrow represents a contact burst. Among them, six contact bursts start around 1pm and end around 2pm. Then, they can form a connected graph, and are most likely from the same TC with duration around $[1, 2]$ pm. On the other hand, other four contact bursts in the example are with different contact durations, and are thus not in this TC. Although this example is simple, in a complex network where contact bursts may not have the exact same starting and ending time, we need techniques to measure their similarity.

3) The Similarity of Contact Bursts: In order to detect TC, our first objective is to find out contact bursts with similar time periods.

Definition 2: The ***similarity of two contact bursts*** $T_{B1} = [t_{s1}, t_{e1}]$ and $T_{B2} = [t_{s2}, t_{e2}]$ is defined as the *Jaccard similarity coefficient*

$$S(T_{B1}, T_{B2}) = \frac{T_{B1} \cap T_{B2}}{T_{B1} \cup T_{B2}} = \begin{cases} \frac{\min(t_{e1},t_{e2}) - \max(t_{s1},t_{s2})}{\max(t_{e1},t_{e2}) - \min(t_{s1},t_{s2})}, & \text{if } \min(t_{e1}, t_{e2}) > \max(t_{s1}, t_{s2}) \\ 0, & \text{if } \min(t_{e1}, t_{e2}) \leq \max(t_{s1}, t_{s2}) \end{cases}$$

Jaccard similarity coefficient is commonly used to measure the similarity between two equal length "0-1" sequences; i.e., the total number of positions where both sequences have 1 divided by the total number of positions where at least one sequence have 1. Similar technique can be applied to our case when two time intervals are compared. The *Jaccard similarity coefficient* for two time intervals are the intersection between them divided by the union of them.

B. TC Detection with CCM

Based on contact bursts and their similarity, we cluster similar bursts together, from which we find TCs. There are many clustering methods in the literature, like *K-means* [22], *spectral clustering* [23] and *hierarchical clustering* [24]. All these methods are aimed to cluster a set of subjects, with the similarity well defined. Using *K-means* algorithm, the number of clusters should be pre-known before clustering. *Spectral clustering* partitions nodes into clusters by using the eigenvectors of a pairwise similarity matrix. Among them, hierarchical clustering is effective and efficient with only one parameter – minimum allowed similarity (γ) that can be flexibly set. Thus, our CCM is based on hierarchical clustering.

With a set of n contact bursts, CCM runs as follows:

1) **Initialization**: Each of the n contact bursts starts to form its own cluster.
2) **Merge clusters**: Pick two clusters with the largest similarity and merge them together. The similarity of two clusters is defined to be the average of all pairwise *Jaccard similarity coefficient* of the contact bursts in these two clusters. This step repeats until the termination condition is satisfied, as defined in the next step.
3) **Termination**: The algorithm terminates if the largest similarity between all clusters in one round is smaller than the minimum allowed similarity γ.

In order for the algorithm to generate TCs, we need to modify the cluster merging phase. With this algorithm, two clusters with similar happening time may be merged as a TC. However, this is not always true. For example, if two clusters don't have any common node, it is more likely that they are

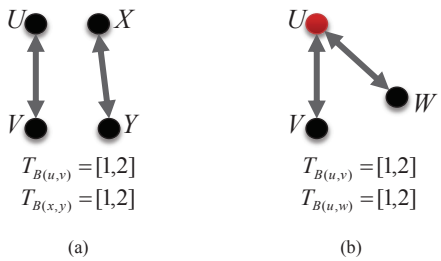


Fig. 5. For two clusters to merge, they must share at least one common node. (a) No common node, nodes U, V and nodes X, Y may come from two different TCs which appear at similar time. (b) Two contact bursts with common node U, and these three nodes form a TC.

from two TCs which happen at similar time but at different locations. Thus, we add one condition for cluster merging; that is, the picked clusters must have at least one common node (as shown in Figure 5). For all pairs that satisfy this condition, those with the largest similarity are merged.

With CCM, contact bursts with similar time periods are clustered together. One cluster corresponds to one TC, where nodes attached to the contact bursts will be in this TC. When the algorithm terminates, there may be some "tiny" clusters, which only include one contact burst. One contact burst only consists of two nodes, and it is more like a personal meeting rather than a TC. For a contact burst $[t_s, t_e]$, its duration is defined as $t_e - t_s$. For a cluster which includes multiple contact bursts, its duration is defined as the average of the durations of these contact bursts. A short cluster duration may represent an occasional encountering of several nodes, which should not be counted as a community. Thus, we should delete clusters that only have one contact burst, and delete clusters with durations shorter than T_{min}, which is a parameter and we set $T_{min} = 0.25$ hour.

Parameter γ: The minimum allowed similarity γ will have impacts on the number of TCs and their average size. We have run CCM on the UCSD trace to see the impacts of γ, and the results are shown in Figure 6. As can be seen, decreasing γ results in a smaller number of TCs and a larger size. When γ is small, the minimum allowed similarity between clusters becomes smaller, and then more clusters will be merged, resulting in fewer TCs with larger size. If γ is too large, the clustering process ends quickly, and contact bursts that should belong to the same TC may not be clustered before termination. Thus, it is important to find the right value for γ.

In a mobile network, contacts that happen between members in the same community are called *intra-community contacts*, and contacts that happen between members in different communities are called *inter-community contacts*. A community structure is preferred if there are more intra-community contacts and less inter-community contacts. Since TC also has temporal information, a contact is called an intra-community contact in a TC only when the contact happens between members in the same TC and the contact happens during this TC's existing period.

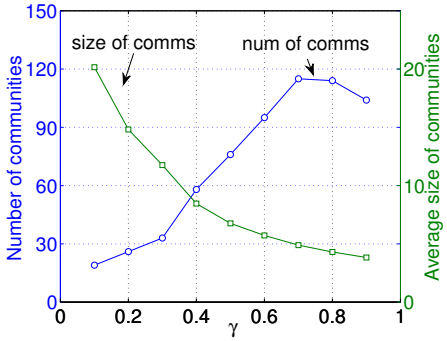


Fig. 6. The number of communities (left, blue) and the average size of the communities (right, green) with varying γ (UCSD trace)

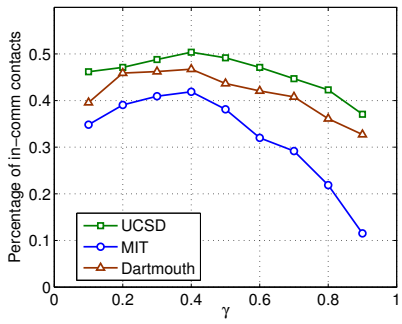


Fig. 7. Proportion of intra-community contacts with varying γ

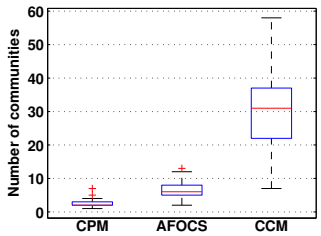
To obtain the percentage of intra-community contacts, we re-run the experiment with the TCs' information and count the number of intra-community contacts. The results are normalized by dividing them by the total number of contacts. The percentages of intra-community contacts in a network using different TC threshold γ are shown in Figure 7, and a larger percentage is preferred. As can be seen, the percentage of intra-community contact is the largest when γ is 0.4, and we choose $\gamma = 0.4$ in the rest of the paper.

C. Evaluations

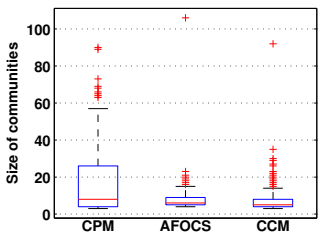
We compare the performance of our TC detection algorithm CCM with existing commonly used community detection algorithms: CPM (K-clique) and AFOCS. The performance is compared based on six metrics, covering various properties of community. Here, we only show the results based on the UCSD trace, since results on other traces are similar.

Number of communities and community size: Figure 8 (a) shows that CCM can detect much more TCs than communities detected by CPM and AFOCS. This indicates that CPM and AFOCS may have mixed some TCs to one community. Figure 8 (b) shows that communities detected by CPM are usually bigger than those detected by AFOCS and CCM. This further confirms that CPM may have mixed some TCs, making the overall community size larger.

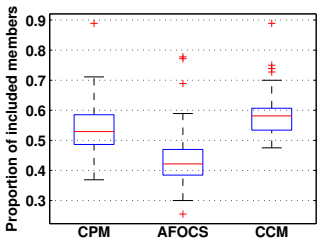
Proportion of nodes involved in community: Community structure is usually used to help data forwarding in DTN; thus it is preferable if more nodes can be included in the



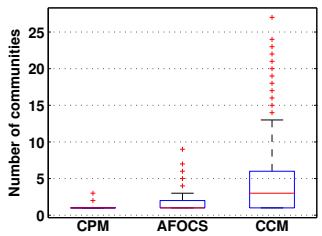
(a) Number of communities



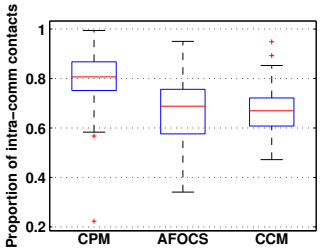
(b) Size of communities



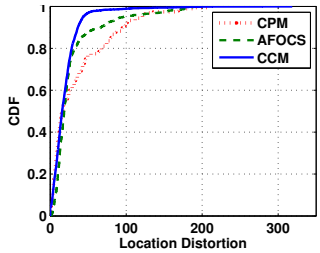
(c) Percentage of users in communities



(d) Number of associated communities for each user



(e) Proportion of intra-community contacts



(f) Location distortion

Fig. 8. Comparisons among CMP, AFOCS and our algorithm CCM

community. Figure 8 (c) shows that CCM usually involves more nodes than CPM and AFOCS. In CPM and AFOCS, the node that has infrequent contacts with others may be ignored.

Number of associated communities for one user: We use this metric to show how much communities are overlapped. From Figure 8 (d), we can see that a node is normally attached to one community in CPM or AFOCS. This means communities are almost mutually disjoint, which does not achieve the objective of detecting overlapping communities. However, TCs detected in CCM have good overlapping property where a node belongs to an average of three TCs. Because TCs are detected using temporal information, it has no limitation on how much communities overlap. Other methods have limits on how much two communities can overlap.

Proportion of intra-community contacts: We prefer a community structure which can incorporate more contacts. For CPM and AFOCS, a contact is considered as an intra-community contact when it appears between two members in the same community. We need another temporal requirement for a contact in TC to be considered as an intra-community contact; that is, it must happen within the community's duration. Therefore, we can see that TC incorporates less intra-community contacts than CPM and AFOCS as shown in Figure 8 (e). CPM has the highest number of intra-community contacts. This can be explained by the fact that there are many large communities detected by CPM, which makes more contacts counted as intra-community contacts.

Location distortion: It measures how much is the difference among nodes' locations within a community. Intuitively, a smaller location distortion means that users in the same community are near each other. On the contrary, a larger location distortion means that users may appear at multiple places in this community. A community's location distortion is the standard deviation of the locations of all intra-community contacts. A contact's location is calculated by averaging two node locations at the contact time.

$$Loc_{<i,j>} = \frac{Loc_{i,t_{<i,j>}} + Loc_{j,t_{<i,j>}}}{2}$$

A community's mean location is calculated by averaging the locations of all intra-community contacts.

$$Loc(C) = \frac{\sum_{<i,j>\in C} Loc_{<i,j>}}{\sum_{<i,j>\in C} 1}$$

The *location distortion* within a community C is defined as the standard deviation of all intra-community contacts' positions.

$$\begin{aligned} Distortion(C) &= \text{stdev}\{Loc_{<i,j>}\} \quad (<i,j>\in C) \\ &= \sqrt{\text{var}\{Loc_{<i,j>}\}} \quad (<i,j>\in C) \\ &= \sqrt{\frac{\sum_{<i,j>\in C}(Loc_{<i,j>} - Loc(C))^2}{\sum_{<i,j>\in C} 1}} \end{aligned}$$

The UCSD trace does not record users' location information, so we estimate user's locations through the detected APs' locations at that time. From Figure 8 (f), we can observe that the location distortion in TCs detected by CCM is smaller than that by CPM and AFOCS; i.e., nodes usually gather at one place in a TC. In a traditional community-based data forwarding approach, data is intended to be forwarded to the communities that include the destination node. In such an application, a smaller location distortion is preferred, because this means nodes are near to each other in one community. Once the data reaches the destination community, it must be near the destination node. In communities detected by CPM and AFOCS, nodes do not have a gathering period, so the contacts between them can appear at any time and any place, which results in a large location distortion.

D. Periodic Appearance of Transient Communities

With the proposed CCM, we can detect TCs on a daily basis. We run the algorithm based on the traces and find that there are many TCs sharing similar members and appearing at different times. These TCs represent one social group which appears periodically, like a class. In addition to these periodically appearing TCs, the majority of TCs only appear once. Such

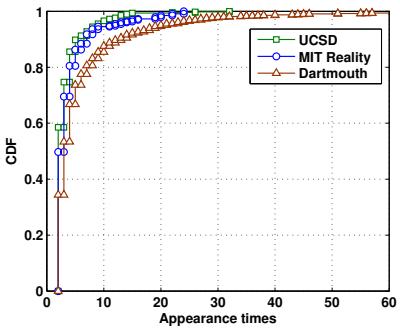
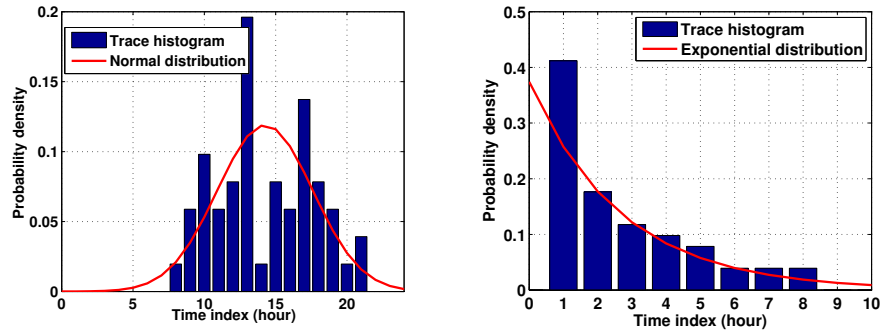


Fig. 9. CDF of TCs' appearance time.



(a) Start time (b) Duration

Fig. 10. The PDF of a TC's start time on a daily basis and the PDF of the TC's duration

randomly appearing communities are usually formed due to the opportunistic meeting of unfamiliar nodes. Later, we will no longer consider these opportunistically formed TCs. In this section, we will identify the periodic appearance of TCs based on the traces and exploit their appearance patterns.

1) Identifying the periodic appearances of TCs: The algorithm we use to cluster similar TCs is the same hierarchical clustering as we presented in Section II-B. TCs are clustered according to the similarity of their members, which is also based on the *Jaccard similarity coefficient*.

We are interested in knowing how many times each TC appears. We run the clustering algorithm on three traces, each with 75 days of data. The CDFs of TCs' appearance times are shown in Figure 9. TCs that only appear once are not included in the figure. As can be seen, there are more than 10% TCs in the Dartmouth trace and about 5% TCs in the MIT and UCSD traces which appear more than 10 times. Meanwhile, there are also some TCs appear more than 30 or 50 times. Given the frequent appearances of TCs, the next question is how to find the appearance patterns of TCs.

2) Appearance patterns of TCs: In this subsection, we study the periodic appearance patterns of TCs on three sets of traces. We formulate the appearance patterns of TCs on daily basis, based on the fact that many social groups are formed daily such as families, offices, and classes. Although there are many social groups formed on a weekly or monthly base or on a more complex pattern, we will not consider these patterns here and leave them as future work. The appearance pattern is formulated in two aspects. One is the distribution of the TC starting time, and the other is the distribution of TC duration.

We have two observations. First, the starting time of a TC within a day can be well approximated by a normal distribution. We take one TC in the Dartmouth trace as an example, shown in Figure 10 (a), and the parameters of the distribution is shown in Table III. The TC usually appears at around 2 pm. Second, the duration of a TC can be approximated by an exponential distribution, as shown in Figure 10 (b). The parameter λ is 0.374, which means that the TC has an average duration of $1/\lambda = 2.674$ hours. In our traces, a TC only appears with limited number of times, and hence the samples used to train the distributions are limited. Therefore, the approximation does not seem perfect, especially for the normal distribution. We believe the approximation should be better if more data are used.

TABLE III
THE PARAMETERS OF NORMAL AND EXPONENTIAL DISTRIBUTION

normal(μ,σ^2)		exponential(λ)
$\mu = 14.256$	$\sigma = 3.356$	$\lambda = 0.374$

Determining the probability of TC appearance within a time interval: Given a time interval $[t_s, t_e]$, we are interested in determining the probability that a TC will appear. It is affected by three factors: the probability that the TC appears in that day, the distribution of its start time in that day, and the distribution of its duration. The probability that a TC appears in a day is simply calculated as the percentage of days when the TC has appeared during the warm-up period. Based on this, whether the TC appears within the pre-defined interval includes two possibilities. The first possibility is that the TC starts in the time period $[t_s, t_e]$, and the second is that the TC starts before the time interval but lasts until the start of the time interval. Suppose the start time is represented by a normal distribution with parameters μ and σ, and the duration is represented by the exponential distribution with parameter λ. The probability of the first possibility is:

$$\begin{aligned} P_1[t_s, t_e] &= \int_{t_s}^{t_e} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt \\ &= normcdf(t_e, \mu, \sigma^2) - normcdf(t_s, \mu, \sigma^2) \end{aligned}$$

where $normcdf(t, \mu, \sigma^2)$ is the CDF of the normal distribution with mean μ and standard deviation σ. The probability of the second possibility is:

$$\begin{aligned} P_2[t_s, t_e] &= \int_0^{t_s} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} * e^{-\lambda(t_s - t)} dt \\ &= e^{-\lambda(t_s-\mu)+\frac{\lambda^2\sigma^2}{2}} \int_0^{t_s} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu-\lambda\sigma^2)^2}{2\sigma^2}} dt \end{aligned}$$

The part under the integral is actually the probability density function of the distribution $norm(\mu + \lambda\sigma^2, \sigma^2)$. Thus, the integral can be easily computed by $normcdf(T_1, \mu + \lambda\sigma^2, \sigma^2) - normcdf(0, \mu + \lambda\sigma^2, \sigma^2)$.

Then, the probability of the TC appearance within this time interval is:

$$P_{TC}[t_s, t_e] \quad = \quad p_d * (P_1[t_s, t_e] + P_2[t_s, t_e]) \qquad (1)$$

where p_d is the probability that the TC appears in the day.

III. Application to Data Forwarding in DTN

Community has been widely used for data forwarding in DTN. However, ignoring the community appearance time may lead to non-optimal forwarding paths. For example, in community-based data forwarding, data is always intended to be forwarded to a node within the destination's community. This is not optimal when considering two problems. First, because traditional community usually has a large location distortion, delivering data to destination's community does not mean it is getting closer to the destination node. Second, considering the appearance time of communities, some of the destination communities to which data are forwarded may not even appear before data expire. Our *TC-based data forwarding strategy* solves these problems by utilizing TCs as the forwarding unit and always forwarding data to TCs that have a better capability of relaying data to the destination node within a short time constraint.

A. Relaying Capability

By estimating the current TC's capability of forwarding data to the destination node within a future time period $[t, t+T]$, we can choose users in TCs with better forwarding capability as the data carriers. We denote the destination node as N_d and the current TC as TC_c. The destination TCs that N_d belongs to are denoted as $TC_d^1, ..., TC_d^n$. Specifically, we first compute the relaying capability of TC_c to each of $TC_d^1, ..., TC_d^n$ within $[t, t+T]$ respectively. Then we can obtain the relaying capability from TC_c to N_d by summing the computed relaying capabilities together.

We next discuss in detail how to compute the relaying capability from TC_c to one of the destination TCs (TC_d^i). The relaying capability from TC_c to TC_d^i within the time period $[t, t+T]$ is computed by summing the probability that each node of TC_c will appear in TC_d^i in $[t, t+T]$. It is determined by the number of common nodes k between them and the probability that TC_d^i will appear in $[t, t+T]$. When we compute the relaying capability from TC_c to TC_d^i within time period $[t, t+T]$, it is implied that TC_c has appeared at time t. The probability that TC_d^i will appear before $t+T$, denoted as $P_{TC_d^i}[t, t+T]$, is computed in Equation (1). The relaying capability from TC_c to TC_d^i is computed as:

$$R_{TC_c \to TC_d^i}[t, t+T] = k * P_{TC_d^i}[t, t+T] \qquad (2)$$

The relaying capability from the current TC TC_c to the destination node N_d is decided by accumulating the relaying capability from TC_c to all the destination TCs N_d belongs to, which are $TC_d^1, ..., TC_d^n$. The relaying capability from TC_s to N_d is computed as:

$$R_{TC_c \to N_d}[t, t+T] = \sum_{i=1}^{n} R_{TC_c \to TC_d^i}[t, t+T] \qquad (3)$$

B. TC-based Data Forwarding Strategy

In this paper, we focus on unicast in DTN, where the source node N_s and the data initialization time T_{in}, the destination node N_d and data's expiration time T_{ex} are given. The objective is to forward the data item from N_s to N_d within the time period $[T_{in}, T_{ex}]$. Our TC-based data forwarding strategy is based on the relaying capability of TCs in the recent time period $[t, t+T]$. That is, our strategy always forwards data to a TC that has better relaying capability. Forwarding decisions are made upon node contacts. If a node meets another node which is in a TC that has larger relaying capability to destination, data will be forwarded. Since whether to forward data depends on the current TC it belongs to, it's important for the node to know which TC it's currently in.

1) Finding the Current TC: To keep track of the TCs that a node is in, each node keeps a queue (Q_w) of the recently met nodes and the contact time within the last time window (T_w). T_w is a constant and how to set the value of T_w will be discussed in Section III-D2. Based on Q_w and all TCs that it has ever known, a matching score is assigned to each TC. The node will consider the TC with the largest matching score as the TC it is currently in. The matching score is computed as follows:

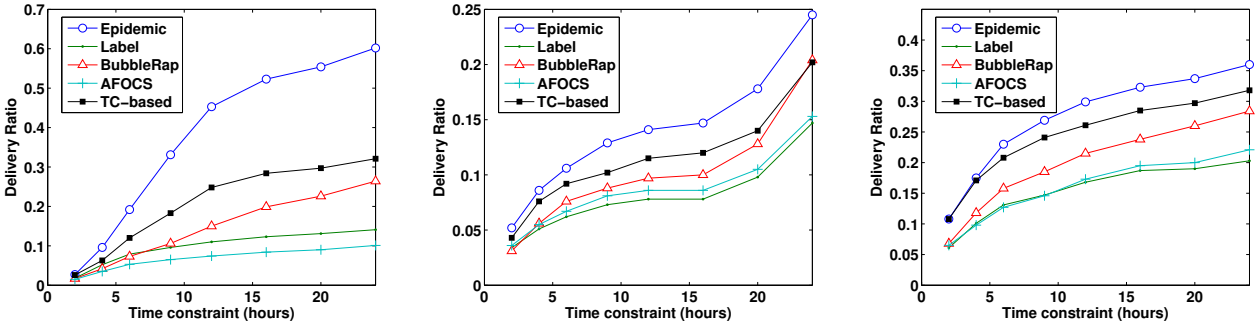
$$Score(TC) = \frac{\sum_{i \in Q_w \wedge i \in TC}(T_w - (t - t_i))}{|TC|} \qquad (4)$$

where t is the current time, and t_i is the node's contact time with the i-th node in queue.

2) Data Forwarding based on TCs: As TC is the data forwarding unit, data is always forwarded to the TC with better relaying capability to the destination. Therefore, once data reaches a new TC with larger relaying capability, the data is distributed to all nodes met in the TC. However, nodes do not carry the data permanently, and they only carry the data for a time period T. If it no longer contacts any node with larger relaying capability or it reaches a TC with larger relaying capability, the data will be deleted at the end of T. The detailed data forwarding process is shown in the following steps:

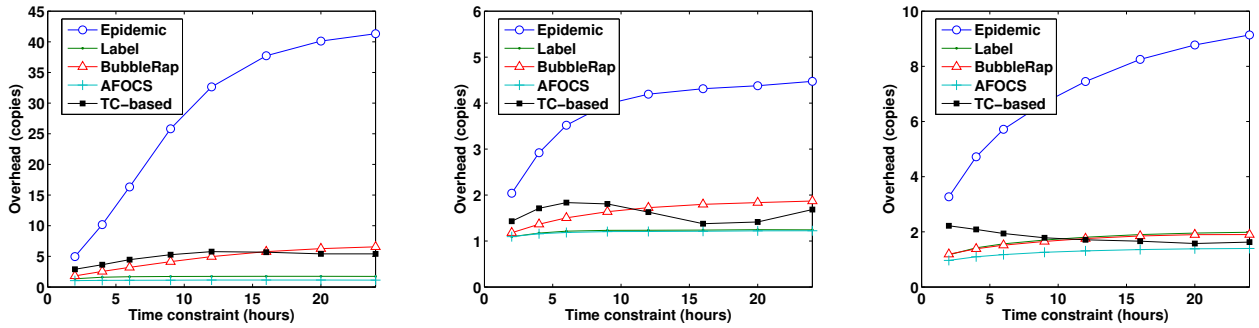
When node N_1 contacts node N_2 at time t,

1) Update Q_w for N_1, and find the TC that N_1 is currently in: TC_1.
2) For each data that N_1 carries.
   a) Check if the data should be deleted from N_1's buffer. The data should be deleted when the node has neither contacted a node in TC with larger relaying capability nor gone to a TC with larger relaying capability in the period $[t-T, t]$. Once the data item is deleted, check the next data item.
   b) Check if the data is carried by N_2. If so, skip this data item.
   c) Check N_2's current TC, TC_2. If TC_2 and TC_1 are the same, N_1 forwards data to N_2 so that the data can be distributed in the current TC. If TC_2 has better relaying capability to N_d, N_1 also forwards data to N_2.



(a) Dartmouth (b) MIT (c) UCSD

Fig. 11. Data delivery ratio of various methods based on three traces



(a)Dartmouth (b) MIT (c) UCSD

Fig. 12. Data forwarding overhead measured by the number of data copies

C. Performance Evaluations

We evaluate the performance of the TC-based data forwarding strategy with the Dartmouth trace, the MIT Reality trace, and the UCSD trace. The value of the time period T, with which we evaluate TCs' relaying capability and decide when to delete data, is empirically set at each trace. We set $T = 1$ hour in all three traces with which we can achieve high delivery ratio with small number of data copies as can be seen in the following results. For each trace, we use the first half as the training period to detect communities and TCs and TCs' appearing patterns, and use the second half of the trace to evaluate the performance. For each data item, sources and destinations are picked randomly and the generation time is randomly chosen in the daytime, since nodes' activity remains low at night which may results in inaccurate comparison. Each experiment is repeated 1000 times for statistical convergence.

Our TC-based data forwarding approach is compared with three traditional community-based forwarding approaches and the Epidemic approach which serves as the upper bound. A brief overview of these approaches is shown below:

- **Epidemic** [25]**:** The data item is always forwarded to another node if it does not have the data. This method has the best data delivery ratio and the highest network overhead, which is used as an upper bound for comparison.
- **Label** [6]**:** This strategy is the first proposed community-based data forwarding strategy. In this strategy, the data item is forwarded to nodes that are within at least one common community with the destination node. CPM (K-clique) is used to detect communities.
- **Bubble Rap** [5]**:** This strategy uses both centrality and community. CPM (K-clique) is used to detect communities. The data item is always forwarded to a higher centrality node, until it reaches a node that belongs to the same community as the destination node. When the data item reaches the destination community, it is forwarded to higher-centrality node within the community's scope, until the destination node is reached.
- **AFOCS:** The strategy was proposed in [7], and it is used to evaluate the communities detected by the AFOCS method. This method is based on how many common communities a node has with the destination node. The data item is only forwarded when the contacted node has more common communities with the destination node than the original carrier.

The performance is measured with two metrics: one is the data delivery ratio and the other is the network overhead. Data delivery ratio is the proportion of data items successfully delivered before the data expires. The network overhead is the average number of data copies existing in the network at each moment. The results are shown in Figure 11 and Figure 12. Generally speaking, our TC-based method has much better performance than the other three community-based methods by achieving much larger delivery ratio with comparable network overhead.

From Figure 11 we can see TC-based strategy consistently

achieves better performance than the other community-based data forwarding algorithms and even get comparable delivery ratio with Epidemic when the time constraint is small. The good performance in small time constraints is due to the fact that we have studied and predicted users' behavior in TCs within a short time period T and forward data to nodes in TCs that have good relaying capability to destination within T.

Figure 12 shows the overhead generated by each approach. Epidemic always has the highest overhead among all the approaches. Our TC-based strategy generates comparable overhead as other three community-based approaches. Since the TC-based strategy removes data when there is no better TC formed or contacted within T, the network overhead is kept low.

D. Discussions

1) Effect of Prediction on Performance: In our TC-based data forwarding strategy, predicting when and whether a TC will happen in the future is important. With prediction, we can compute the relaying capability of each TC to the destination node and then use the nodes in TCs with good relaying capability to carry data. To evaluate how the prediction affects the performance, we compare our strategy with a TC-based strategy without prediction (i.e., only distributing data when a node finds that the contacted node or itself is within the same TC of the destination node). Figure 13 compares their data delivery ratios. We can clearly see the superiority of the strategy with prediction.

2) Effect of Time Window T_w: In Section III-B1, we used the information of recently met nodes in the last time window T_w to identify the current TC. To find the optimal value of T_w in each trace, we conduct an experiment to see how T_w impacts the data delivery ratio. Here, the data delivery ratio is recorded with a time constraint of 12 hours. The results are shown in Table IV. As can be seen, $T_w = 0.5$ hour has the best data delivery ratio for all traces.

TABLE IV
EFFECT OF T_w ON DATA DELIVERY RATIO

T_w	0.5	1	2
Dartmouth	0.248	0.217	0.205
MIT	0.115	0.114	0.112
UCSD	0.261	0.26	0.26

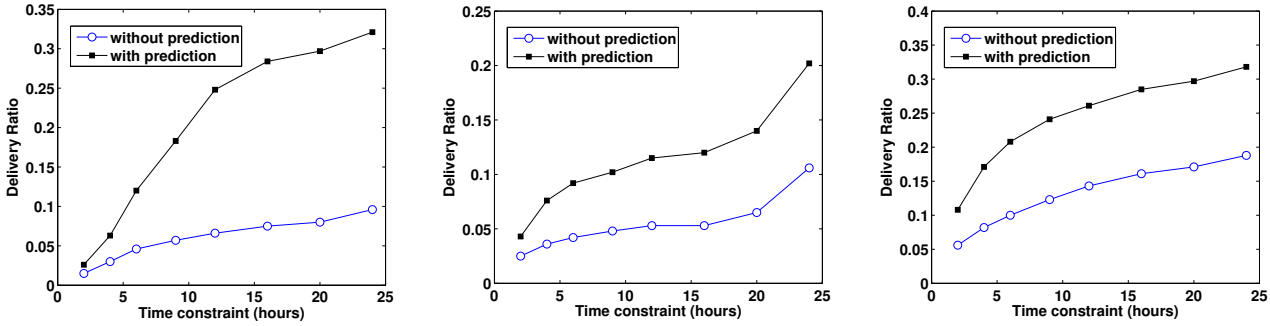
3) A Distributed Strategy: Our TC-based strategy can be run in a distributed manner. Since the CCM algorithm must be run in a centralized manner, we centrally detect all the possible TCs in the training period and distribute the TCs' information to each node. In the experiment, a node is able to identify its current TC in a distributed manner with the information of recently met nodes using the method introduced in Section III-B1. The TC-based data forwarding decision can also be made by individual node upon contacts, with the forwarding process discussed in Section III-B2. Since TCs may not be stable, we can detect the TCs again at the end of a fixed period (like one month). Then, the updated TC information can be redistributed.

IV. RELATED WORK

Complex networks usually consist of communities. Lots of research has been done on detecting communities by interdisciplinary researchers from physics, biology and computer science. They originally aimed at identifying communities in a static network. There are many methods focused on detecting disjoint communities such as the modularity-based methods [26][15]. Techniques have also been proposed to detect overlapping communities such as CPM [18] and AFOCS [7], and a survey of existing community detection methods is given in [13]. Although community detection has been well studied in static networks, it remains difficult to detect time-varying communities in a dynamic network such as a social network in which people's behavior is highly dynamic. Recently, there is more research on years have seen a trend for studying evolving community structures based on the network structures at successive network snapshots. Palla *et al.* [27] developed an algorithm to identify evolving communities by first detecting overlapping community structure at each network snapshots and then mapping similar communities at different snapshots. Nguyen *et al.* [7] proposed AFOCS, which can adaptively update the community structure at each network snapshot based on history without detecting communities again at each snapshot. These methods are intended to analyze the long-term evolution of community structures due to the permanent change of humans' behaviors or habits. There is still a dearth of work examining transient communities caused by the periodic change of human behavior.

Although Wei *et al.* proposed the concept of transient community in [3], they only investigated the community relationship between individual pairs of nodes without providing the complete knowledge about the transient community structure. Pietilanen *et al.* [28] proposed algorithms to detect temporal communities that are similar to transient communities. The basic idea is to extract static community structure from each network snapshot. However, the time interval of the network snapshot is hard to determine; thus it is hard to accurately detect transient communities. Our work is not based on network snapshot but a careful study of the pairwise nodes' contact processes, and thus can accurately detect transient communities.

Community structure has been extensively utilized to address the problem of data forwarding in DTN. It is believed that nodes within the same community have higher chance to contact each other. Pan *et al.* in [6] and [5] have proposed to forward data to the nodes that are within the destination's community based on the static community structure. By introducing dynamic community structure, AFOCS [7] further considered community evolution in the community-based data forwarding. Transient communities in [3] were only used to determine the scope for evaluating node centrality. [28] studied how temporal communities contribute to data dissemination and concluded that temporal communities tend to limit data



(a)Dartmouth (b)MIT (c)UCSD

Fig. 13. The impact of prediction in the TC-based data forwarding approach.

dissemination in DTN. As far as we know, our paper is the first work to fully utilize transient communities for data forwarding.

V. Conclusions

In this paper, we proposed a contact-burst-based clustering method (CCM) to detect TCs by exploiting the pairwise contact processes. We formulated each pairwise contact process as regular appearances of contact bursts during which most contacts between the pair of nodes appear. Based on such formulation, we detect transient communities by clustering the pairs of nodes with similar contact bursts. Trace-driven simulations showed that CCM can detect TCs more effectively compared with existing community detection methods. We also applied the concept of TCs to data forwarding in DTN, where data is forwarded to TCs that have better relaying capability to the destination node. Trace-drive simulations show that our approach outperforms traditional community-based data forwarding approaches.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. ACM SIGCOMM*. ACM, 2003, pp. 27–34.

[2] E. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proc. ACM MobiHoc*. ACM, 2007, pp. 32–40.

[3] W. Gao, G. Cao, T. La Porta, and J. Han, "On exploiting transient social contact patterns for data forwarding in delay tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 151–165, 2013.

[4] W. Gao and G. Cao, "User-centric data dissemination in disruption tolerant networks," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 3119–3127.

[5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 11, pp. 1576–1589, 2011.

[6] P. Hui and J. Crowcroft, "How small labels create big improvements," in *PerCom Workshops*. IEEE, 2007, pp. 65–70.

[7] N. Nguyen, T. Dinh, S. Tokala, and M. Thai, "Overlapping communities in dynamic networks: their detection and mobile applications," in *Proc. ACM MobiCom*. ACM, 2011, pp. 85–96.

[8] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *Pro ACM MobiHoc*. ACM, 2009, pp. 299–308.

[9] M. McPherson, L. Smith-Lovin, and J. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[10] B. Han and A. Srinivasan, "Your friends have more friends than you do: identifying influential mobile users through random walks," in *Proc. ACM MobiHoc*. ACM, 2012, pp. 5–14.

[11] Y. Zhang, W. Gao, G. Cao, T. La Porta, B. Krishnamachari, and A. Iyengar, "Social-aware data diffusion in delay tolerant manets," in *Handbook of Optimization in Complex Networks: Communication and Social Networks*. Springer, 2012, pp. 457–481.

[12] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, "A social network based patching scheme for worm containment in cellular networks," in *INFOCOM 2009, IEEE*, 2009, pp. 1476–1484.

[13] S. Fortunato and C. Castellano, "Community structure in graphs," *arXiv preprint arXiv:0712.2716*, 2007.

[14] S. Asur, S. Parthasarathy, and D. Ucar, "An event-based framework for characterizing the evolutionary behavior of interaction graphs," in *Proc. ACM SIGKDD*. ACM, 2007, pp. 913–921.

[15] M. Newman, "Analysis of weighted networks," *Physical Review E*, vol. 70, no. 5, p. 056131, 2004.

[16] Z. Lu, Y. Wen, and G. Cao, "Community detection in weighted networks: Algorithms and applications," in *PerCom 2013, IEEE*, 2013, pp. 179–184.

[17] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[18] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[19] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. ACM MobiCom*. ACM, 2004, pp. 187–201.

[20] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.

[21] M. McNett and G. Voelker, "Access and mobility of wireless pda users," *ACM SIGMOBILE CCR*, vol. 9, no. 2, pp. 40–55, 2005.

[22] J. Hartigan and M. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[23] I. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts," in *Proc. ACM SIGKDD*. ACM, 2004, pp. 551–556.

[24] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer Series in Statistics, 2001, vol. 1.

[25] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[26] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[27] G. Palla, A. Barabasi, and T. Vicsek, "Quantifying social group evolution," *Nature*, vol. 446, no. 7136, pp. 664–667, 2007.

[28] A. Pietilänen and C. Diot, "Dissemination in opportunistic social networks: the role of temporal communities," in *Proc. ACM MobiHoc*. ACM, 2012, pp. 165–174.