2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems

CLAN: An Efficient Distributed Temporal Community Detection Protocol for MANETs

Stephen Dabideen, Vikas Kawadia, and Samuel C. Nelson
{dabideen, vkawadia, snelson}@bbn.com
Raytheon BBN Technologies, Cambridge, MA, USA

***Abstract*—Real world MANETs often exhibit an inherent community structure in their topological connectivity and in the evolution of the topology over time. Such temporal community structure of MANETs has been shown to be extremely useful in improving the performance of routing and content-based routing in MANETs [1]–[4]. However, detecting temporal communities in a completely distributed and real time manner is a hard problem, and it is often performed offline with knowledge of the full network topology over time.**

We propose CLAN, a distributed and real-time protocol for detecting temporal communities in MANETs. CLAN is an adaptation of the Label Propagation algorithm [5] to distributed and time-varying graphs that MANETs are. A key novel component of CLAN is local rules for community rediscovery as the network evolves. CLAN also uses a weighted version of the network topology where the weights are defined using a novel notion of social entropy to promote stability of communities. Extensive simulation results demonstrate that CLAN is quick to converge, incurs minimal overhead and is as effective as centralized approaches to temporal community detection. We also demonstrate how the temporal community structure can be used by designing a hierarchical routing protocol that achieves the delivery ratio of the OLSR [6] routing protocol at a fraction of the overhead.

I. Introduction

Almost every smartphone is now capable of direct phone-to-phone WiFi communication using WiFi-Direct, Bluetooth 3.0/4.0, and in some cases the 802.11 ad hoc mode. Thus, large deployed MANETs now exist in the real world. Applications to take advantage of this huge potentially planet-scale MANET are also beginning to emerge. Examples include FireChat, WiFi Social, HitcherNet etc. Many of the applications are however still single hop as scaling MANETs to many hops and to significant traffic remains a challenge. A rich trove of routing protocols have been proposed over the years for improving MANET scalability, and more recently content-centric MANETs have also been designed and built [7]. In this paper, we look at a fundamental building block for building scalable MANETs: *community detection*.

We argue that connectivity in many mobile ad-hoc networks, particularly those exhibiting human-centric mobility patterns (e.g., networks formed from phones carried by humans) exhibit community structure rather than random connectivity. That is, the network can be partitioned into dense groups such that there are more intra-group links than inter-group links. The detection of communities, taking into account both temporal and spatial connectivity, in a distributed fashion, is referred to as *distributed temporal community detection* and is the focus of this paper. The foundations of scalability and efficiency in MANETs lie in exploiting the underlying community structure of a network for routing, aggregation and load-balancing.

Community detection in static graphs has been widely studied and shown to reveal latent yet meaningful patterns in a variety of fields ranging from social networks to systems biology; see the comprehensive survey by Fortunato [8] for more details. Recent work (e.g. [9], [10]) on temporal communities detection extends the notion of communities to time-varying graphs by seeking to detect how communities emerge, grow, merge and split overtime. Intuitively, temporal communities are a partitioning of the network nodes into (relatively) dense groups such that the intra-group connectivity is (relatively) stable over time. Temporal community structure is a basic building block that has been used to improve performance of routing in MANETs, DTNs and content based networks. We describe these applications in detail in Section II.

However, temporal community detection that is distributed, efficient, and fast is especially challenging in a MANET environment. The first challenge is that the global topology information is either unavailable or expensive to maintain. Thus, we must use a local protocol that primarily involves communication with one-hop neighbors. The next challenge is that that time is not apriori divided into snapshots or windows over which we can aggregate node-contacts to construct graphs. We must monitor the network as it is evolving to detect changes in the community structure and adapt to those triggers. The final challenge is to detect stable communities that incorporate temporal structure in node mobility which may differ from the instantaneous connectivity.

In this paper, we present CLAN - Community detection via Label propagation in Ad-hoc Networks. CLAN is a distributed community detection algorithm that relies simply on periodic one-hop broadcasts to neighbors. No global topology knowledge or periodic network snapshots are required. CLAN is based on the Label Propagation Algorithm(LPA) proposed by Raghavan, Albert and Kumara [5] for static (e.g., non-temporal) graphs. In LPA, each node is initially assigned a unique label, which serves as the identifier of the community to which the node belongs. During an iteration of LPA, each node updates its label to the most popular label among its one-hop neighbors. This process continues until the set of labels is stable. CLAN adapts LPA in three key ways: First, it makes

978-1-4799-6036-1/14 $31.00 © 2014 IEEE
DOI 10.1109/MASS.2014.77

LPA distributed. Second, it allows the community structure to adapt to the topology by specifying how nodes "reset" their labels and rediscover the current community structure on specific topological triggers. Third, CLAN specifies a time-varying function to assign weights to links before doing the community detection. This weight assignment helps us discover stable communities by excluding highly mobile nodes (e.g. data mules) from the temporal communities.

We evaluate CLAN thoroughly on real and synthetic traces using the ns3 simulator. Results show that CLAN is able to detect the temporal community structure as accurately as a centralized, offline algorithm in a wide range of scenarios. Furthermore, CLAN is able to do so at a low overhead which makes it appealing as a building block for more complex protocols. We design a novel one ourselves: community-based hierarchical routing (Section V), and show that it achieves the delivery ratio of the OLSR routing protocol at a fraction of the overhead.

The rest of the paper is organized as follows. Section II, overviews existing work on temporal community detection and its applications to wireless networks. Section III describes the CLAN protocol in detail, and Section IV presents the evaluation results.

II. Community Detection Applications and Related Work

Temporal community detection is a widely applicable building block for efficient ad-hoc networks. If accurately done with minimal overhead, its fundamental nature can enhance existing networks and applications, as well as enable new technology and paradigms. These include routing in disruption tolerant networks, routing in pocket-switched social networks with widely varying topological densities, information-centric networking at the edge, and security and privacy in social networks.

Two example applications where temporal community detection is immediately applicable is DTN routing and content-based edge networking. In DTN environments, instantaneous end-to-end paths are not guaranteed to exist. However, due to inherent structure and clustering commonly found in these networks, particularly ones exhibiting human-centric mobility patterns, temporal community detection allows for more intelligent forwarding decisions to be made. A common approach used to deliver messages between humans is to pass the message to a carrier that is known to be in close contact with the destination. This same tactic of "social forwarding" has been successfully applied to social DTNs [3], where a forwarding node prefers to make progress towards nodes within the destination's community. Furthermore, hybrid DTN and MANET environments, with reasonably connected clusters operating together in a disconnected fashion, are not well suited for straight DTN routing or straight MANET routing. With DTN protocols, many replicas are needed to reach nodes multiple hops away [11], [12] which is inefficient for traversing connected clusters. For this reason, protocols have been developed that use a combination of techniques, such as single-copy, MANET-style routing to traverse clusters and DTN-style routing to bridge clusters [13]–[15]. Temporal community detection allows discovery of these clusters, and provides a foundation for managing and distributing data flowing through them.

Temporal community detection also allows for efficient management of pooled network resources within a community, which is critical in content networking protocols with heavy in-network caching. One technique, using a DHT to organize storage space within a particular cluster, has been used to implement efficient content networking solutions for MANETs [4]. In this particular scheme, the "publish" operation places a single copy of the published content in all clusters, using a DHT to determine storage responsibility within the cluster. The "query" operation then fetches the content from the responsible node in its local cluster's DHT. Accurate and efficient community detection is necessary in order to minimize DHT churn and maintain a small number of overall replicas.

Numerous community detection algorithms have been proposed, with the focus ranging from social communities [16] to biological networks [17] to communication networks [18]. Raghavan et. al proposed the Label Propagation Algorithm (LPA) [5], where nodes iteratively adopt the most common label in its neighborhood. Barber and Clark [19] showed that LPA is equivalent to maximizing the Hamiltonian for a ferromagnetic Potts model [20] and that the globally optimal solution is achieved when all nodes have the same label, irregardless of community structure. To solve this problem, they added a constraint to LPA that penalizes undesirable solutions and showed that this constraint can be set so that constrained LPA performs modularity maximization. Modularity maximization [18] involves partitioning nodes so that the fraction of intra-community edges, in excess of that in a random graph, is maximized.

For community detection in time-varying networks, many authors [2], [21], [22] have proposed maximizing some quality function (e.g. modularity) over discrete snapshots of the network and mapping communities across snapshots to the same label. The community structure in each snapshot is independent of that in the previous snapshot, leading to a possibly unstable community structure. Multislice modularity optimization was introduced by Mucha et. al. [23] to extend constrained LPA to time-varying network. This algorithm is centralized and requires snapshots of network connectivity (called slices). Connections, called interslice links, are made between consecutive slices to map nodes across slices. These interslice links allows the coupling of adjacency matrices, allowing the community structure in one slice to influence the detected community structure in the subsequent slice, increasing the stability of the detected community structure over time. This algorithm cannot be applied to a distributed system because of its reliance of discrete snapshots.

Dang et. al. proposed a distributed clustering protocol [24] based on the average pair-wise contact time. In this protocol, clusters are defined such that each node must have some

minimum average contact (γ) time with *every* other node in the cluster. Drugan et. al [25] modify the modularity-based approach of Newman and Grivan [18], the random walk approach proposed by van Dongen [26] and the q-spin Potts model proposed by Reichard and Bornholdt [27], to accept as input the topology table maintained by the OLSR routing protocol [6] rather than a complete graph. The paper demonstrated that the OLSR topology table was sufficient to accurately detect the communities in the network without additional overhead, under the given network conditions. CLAN incurs significantly less overhead than OLSR since we rely primarily on neighbor discovery beacons and not network-wide link state broadcasts.

In [28], the authors present an adaptive algorithm to detect the evolution of overlapping temporal communities. They define a community to be a group of nodes where the number of internal edges, expressed as a fraction of the total possible number of edges between the nodes in the community, is greater than some user-specified threshold. Hui et. al proposed a similar approach using *local modularity*, to partition the network into communities and to guide the incremental evolution of the network. Local modularity is based on the the ratio of intra to inter community edges of the border nodes, which is difficult to compute in a dynamic, distributed environment.

CLAN is a distributed temporal community detection protocol that is distinct from the above approaches in that it does not need global topology information, does not need offline aggregation of the time-varying network topology into snapshots or windows.

III. Protocol

In this section, we describe the CLAN protocol in full detail. CLAN has three basic components: i) a distributed adaptation of the Label Propagation algorithm [5] ii) a mechanism to re-initialize the algorithm based on local triggers whenever sufficient changes in community structure is detected and iii) a scheme to assign weights to links based on contact history and novel notion of social entropy thus allowing the detection of temporal communities rather than instantaneous community structure.

A. Label Propagation Algorithm and its distributed adaptation

The Label Propagation Algorithm (LPA) was proposed by Raghavan, Albert, and Kumara [5] for community detection in static graphs. In LPA, each node is initially assigned a unique label, which serves as the node's community identifier. In every iteration of LPA, each node adopts the most popular label among its one-hop neighbors as its own label. The algorithm terminates when no node changes its label.

Barber and Clark [19] analyzed LPA more formally. Let, ℓ_x denote the label at node x, and $\delta(\ell_u, \ell_v)$ be the the equality indicator between ℓ_u and ℓ_v (i.e., $\delta(\ell_u, \ell_v) = 1$ if $\ell_u = \ell_v$, and equals 0 otherwise). Let A_{uv} be the components of the adjacency matrix (i.e. $A_{uv} = 1$ if u is adjacent to v). Then LPA finds a local minima of the objective function

$$H = \frac{1}{2}\sum_{v=1}^{n}\sum_{u=1}^{n} A_{uv}\delta(\ell_u, \ell_v) \quad (1)$$

$$= \frac{1}{2}\left(\sum_{v\neq x}\sum_{u\neq x} A_{uv}\delta(\ell_u, \ell_v) - A_{xx}\right) \quad (2)$$

$$+ \sum_{u=1}^{n} A_{ux}\delta(\ell_u, \ell_x). \quad (3)$$

In each iteration of LPA, every node x maximizes the second term $\sum_{u=1}^{n} A_{ux}\delta(\ell_u, \ell_x)$, which maximizes its contribution to the overall objective (1) since the first term is independent of ℓ_x. Tibely and Kertesz [20], also show that LPA is equivalent to finding a local minima of a simple Potts model in statistical mechanics.

Our first contribution is a distributed adaptation of LPA. This is relatively straight-forward since each node uses only local information, namely the most recent label of each its one-hop neighbors to decide its current label. In the distributed adaptation, each node recomputes and transmits its label roughly every τ seconds, in it's hello messages. Nodes record the latest label for each of its neighbors, to be used in the next recompilation.

B. Detecting evolving community structure

LPA was designed for static graphs with fixed community structure. After each iteration of LPA, the number of labels decreases or remains the same. It is therefore impossible to detect the emergence of a new community. A naive approach would be to re-initialize LPA by resetting node labels of all nodes to their node identifiers whenever there is any change in the network topology. Reinitialization, however, is expensive and disruptive. Further, in a mobile network, such an algorithm may never converge. To reduce the negative impacts, with CLAN, we investigate reset triggers and localized resets.

Our key contribution is to observe that only a small fraction of topology events require resets, most events do not cause a change in the community structure and thus do not require resets. Moreover, each node can locally determine the events that require resets and the reset can be restricted to the relevant communities. The basic idea is for each node to keep track of the change in its objective function, H, since last convergence of LPA and issue a reset to all nodes in its community when this change crosses a threshold (ΔH). We prove these assertions, demonstrating that CLAN can efficiently detect and discover evolution in community structure over time.

We modify the objective function of LPA (Equation 1), to decrease with the addition to inter-community links by redefining δ: $\delta(\ell_u, \ell_v) = 1$ if $\ell_u = \ell_v$, and equals -1 otherwise. This allow for label resets when communities merge, otherwise LPA may not detect mergers between communities of comparable size.

Lemma 1. *LPA still works when $\delta(\ell_u, \ell_v)$ is redefined to be equal to 1 if $\ell_u = \ell_v$, -1 otherwise.*

Proof. Consider a node u with n neighbors, whose labels are $L_1, L_2, L_3, ., L_n$. Let $|L_x|$ denote the number of neighbors with label x. Let p be the label selected by LPA, i.e. p is the most popular label in A's neighborhood and $max(\sum_{u=1}^{n} A_{uv}\delta(\ell_u, \ell_v)) = |L_p|$.

With the new definition of $\delta(\ell_u, \ell_v)$, $\sum_{u=1}^{n} A_{uv}\delta(\ell_u, \ell_v) = |v| - (n - |v|) = 2|v| - n$, for any label v. This is maximized when $|v|$ is maximized, which happens when the most popular label is chosen, i.e. $v = p$. Therefore, the redefinition of $\delta(\ell_u, \ell_v)$ does not change the label chosen by LPA. □

LPA, with this modified objective function, should therefore uncover static communities in the network. The caveat is that once the algorithm has converged, it must be reinitialized to discover changes in the community structure. We argue that nodes can use changes in its locally computed objective function H to determine when resets are necessary.

Lemma 2. *An increase in the objective function, H, does not indicate a change in community structure.*

Proof. From Equation 3 and the above modified definition of δ, an increase in the objective function occurs when intra-community links are added or inter-community links are deleted. Communities are defined such that there are more intra-community links than inter-community links. This adding intra-community links or removing inter-community links only reinforces the existing community structure and does not correspond to a change in the underlying community structure. □

From the above lemma, resets should only be triggered when there is a decrease in the objective function. The sensitivity parameter, σ, quantifies the change necessary to trigger a reset. A larger value of σ allows for quicker detection in the community structure but also increases the frequency at which resets happen in a mobile network. Empirically, we find that values of $\sigma > 2$ are sufficient to detect changes, and it is up the user to specify the tradeoff in responsiveness and frequency of resets, based on the network. To limit the effects of label resets, we argue that network-wide resets are not necessary but that community-wide resets are sufficient.

Lemma 3. *When the network changes after LPA has converged, community-wide labels resets are sufficient to discover a reasonable community structure.*

Proof. This is simply saying that link changes on a set of nodes only affect the communities that those nodes belong to. If there is not a sufficiently large decrease in the objective function at any node of an established community, then that community still exists in its current form and there is no need to reinitialize LPA regardless of changes elsewhere in the network. □

Conjecture 1. *ΔH is a good local indicator of potential changes in the global community structure.*

Comment An evolution in community structure must correspond to the addition of a significant number of inter-community links (for a merger), or the removal of a significant number of intra-community links) a split. The intuition behind this proposition is that in either case, at least one node in the existing community structure will undergo significant changes in its ΔH.

C. Temporal Communities, Link Weights and Social Entropy

The temporal community structure is defined over a period of time, and may differ from the community structure indicated in a snapshot of instantaneous connectivity. For example, a node may temporarily move to the region of community A, but spends most of its time in community B. Such a node clearly belongs to the temporal community of B, it may belong to the instantaneous community of A. This issue is easily addressed with the use of an exponentially weighted moving average (EWMA) for the link weights. Node updates the EWMA link weight to every other node before sending the periodic hello message. If a *hello* was received in the last interval, the associated link weight is increased, otherwise it is decreased.

We take a novel approach by allowing nodes to individually determine the coefficient, α, of the EWMA. Drawing from social networking, we define the *social entropy* of a node as the rate of change of connectivity of that node with other nodes of the network. Intuitively, nodes with higher social entropy should be less inclined to join a community, thus we make α proportional to the social entropy. We formally define social entropy in Equation 4 and its use in Equation 5.

This notion of social entropy is particularly relevant to MANET as it prevents data mules from joining communities. Data mules, by design, frequently move between communities and will therefore have high social entropy. By Equation 5, the rate at which their link weights increase will be small. Therefore, data mules will be required to stay within the proximity of a community for a longer time than a non-data-mule before it joins that community. As a node moves out of range of a community, the value of its objective function decreases, and it will eventually cause a reset before it completely leaves the community. It is therefore beneficial to differentiate data mules from other nodes in the network. Simulation results in the next section demonstrate that this definition and use of social entropy results in data mules creating and maintaining their own individual communities.

$$S^A(t) = \frac{|N|}{\sum\limits_{B \in N} L^{AB}(t) * \mu^{AB}(t)} \tag{4}$$

$$\alpha^A(t) = \frac{\alpha(0)}{S^A(t)} \tag{5}$$

D. The CLAN Algorithm

In CLAN, nodes can be in any of three states, as illustrated in Figure 1. All nodes are initially in the $unstable$ state and they use CLAN's modified weighted LPA to update their labels every hello interval. If a node does not change its label for $\psi * \tau$, then we assume LPA has converged and the node enters

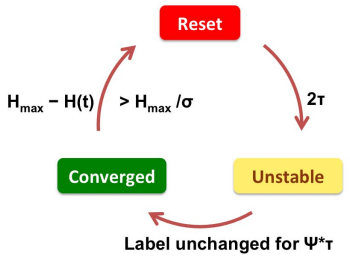


Fig. 1. Flow diagram of LPA

the *converged* state. Once in the converged state, nodes do not run LPA but monitor the difference in the current value of the objective function from its maximum value since the last reset (ΔH). If $\Delta H > H_{max}/\sigma$, and the node's degree is greater than 3, the node initiates a community-wide label reset and enters the reset state. We introduce this minimum degree exception to prevent border nodes from constantly resetting communities. Nodes must wait in the *reset* state for 2τ before re-entering the *unstable* state.

IV. Simulation Results

We evaluate CLAN in NS3 using real and synthetic mobility traces. For comparison, we use multislice modularity optimization [23] and the Louvain algorithm [29]. Like CLAN, the multislice clustering algorithm detects and preserves temporal communities, by finding relationships between nodes over time (slices). However, the multislice algorithm is neither distributed nor online as it requires complete topology snapshots. We create snapshots of the network connectivity every 4 seconds as use these snapshots as input to the centralized algorithms. CLAN is completely distributed and we demonstrate that it is able to detect community structure as accurately as the centralized algorithm.

Our simulation results show that CLAN is able to detect evolution in community structure over time, without excessive label resets. We further demonstrate that CLAN can preserve communities even if multiple communities are collocated for a short period of time, a stark contrast to non-temporal community detection algorithms. More interestingly, CLAN is adaptive so that the social entropy dictates the ease at which communities can evolve, and this changes over time. Consequently, high entropy (e.g. data mules) do not readily join communities. Unless otherwise stated, we set the (σ) in CLAN is set to 4, the convergence threshold ψ is set to 5 and the hello interval (τ) to 1 second. The *interslice* link weight in the multislice algorithm is set to 1 in the multislice algorithm. Empirically, we find this is a good balance between the frequency of resets and the time taken to detect evolution of community structure.

We illustrate the results using *tiled* plots, with time on the x-axis and node ID on the y-axis. If a group of nodes are the same color over a contiguous period of time, they belong to a temporal community for that duration.

A. CBMANET dataset

We use the CBMANET mobility trace [30]–[32] to demonstrate that CLAN is able to detect time-varying communities in real networks. In this trace, there are 40 soldiers, arranged into 6 platoons. Each of these platoons travel one of two paths shown in Figure 2. The platoons move from one staging area to the next, labeled A-E in Figure 2, one platoon at a time. Initially all platoons are at the start point. Platoon 1 moves to point A, then platoon 2 moves to A, followed by platoon 3. While platoon 3 is moving to point A, platoon 1 starts to move to point B. Platoon 1 and 2 are concurrently at each intermediate point for some time and likewise for platoons 2 and 3. However, platoon 1 and 3 are never collocated, except at the start point and the end point. This trace is particularly interesting as the communities merge and split as the platoons meet and separate, demonstrating evolution in the communities over time.

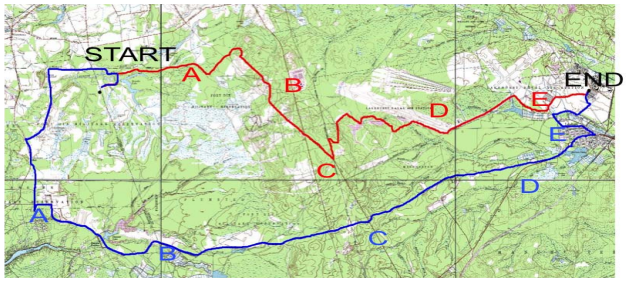
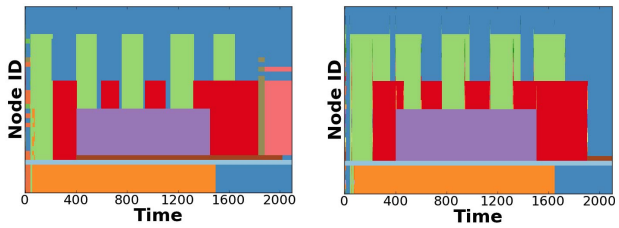


Fig. 2. The paths and intermediate meeting points in the CBMANET mobility trace

From Figure 3(b) , some of the platoons are easily identifiable. Those colored orange, and the small strip of light blue are two platoons that do not merge with other platoons at the intermediate points. The darker blue community, at the top of the plot is platoon 1. It merges with platoon 2 at the intermediate points and they both use the blue color when this happens. The third platoon is red when they are isolated but they join platoon 2 at the intermediate points and adopt the green color of platoon 2 at these points. There is only one community at the start, and all but 2 soldiers are together at the end. It is evident by comparing Figure 3(a) to Figure 3(b) that CLAN is able to detect the same communities as the multislicing algorithm, with more stability towards the end of the experiment.

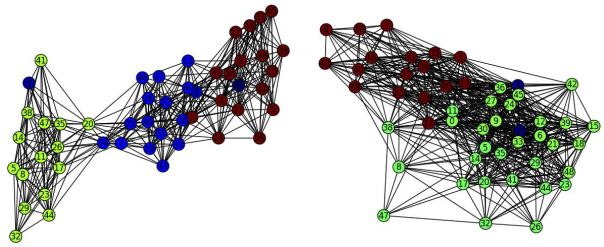
B. Synthetic mobility trace based on social network theory

We use a synthetic mobility generator [33], based on a social networking theory. In this model, the user specifies the number of nodes and the number of communities. Each node is randomly assigned to a community and an *interaction matrix* is generated based on the community assignments. Nodes belonging to the same community are attracted to each other more than to nodes belonging to a different community s. Each community is mapped to a unique geographical space and member nodes move from one point to another within



(a) Multislicing (b) CLAN

Fig. 3. CLAN detects almost the same community structure as the centralized multislicing algorithm.



(a) Connectivity at time t=340s (b) Connectivity at time t=640s

Fig. 4. In this example, there are initially one community, which then splits into three communities and after some time, two of the communities merge to form a single community

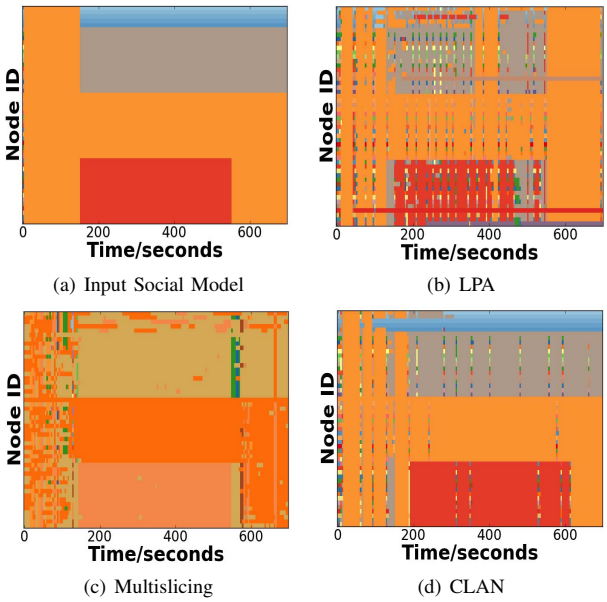
the assigned space. Their destination position is determined by the interaction matrix; nodes move towards nodes they are attracted to. Consequently members of the same community cluster together as they more strongly attracted to each other. However, they do have some attraction to non-community members and can move away from their communities, at least temporarily. Different community structures can be specified for different timescales. As nodes are reassigned, they will be strongly attracted to the geographical region associated with their new communities. The model also allows for travelers, which have roughly the same attraction to every node in the network and move frequently between communities. These travelers can represent city buses or taxis moving around a city and serve as data mules in communication networks. We use an area of 1000m x 1000m and a radio range of 150m. In general, there are links between nodes of different communities but there is even more links between nodes of the same community, as seen in Figure 4 This mobility model has been validated with real traces and it has been shown to be a good approximation of human movement patterns [33].

C. Evolution of Communities over time

We generate two simple traces to demonstrate that CLAN is able to detect basic community evolution: mergers and splits. The first scenario consists of 50 nodes, 5 of which are travelers. As the input to the social model, we start with one community, which splits into three communities at 150 seconds and two of these merge at 550 seconds. After each change in the social model nodes will begin to move to create a topology that reflects the new structure. The movement is not instantaneous and it takes time before the new community structure emerges. In Figure 4 we show snapshots of the connectivity of the nodes at two points in time, with different community structures.

Figure 5(a) shows the input to the social model, which we treat as ground truth. This figure is based solely on community assignment and not on location or connectivity. Each color represents a community, and nodes with the same color at any instant, belong to the same community.

With LPA, community structure is based solely on instantaneous connectivity. If two communities are close enough at any instant, they may merge and this leads to inaccurate community structure as can be seen at the end of the experiment. Multislice and CLAN deliver similar results, as they both preserve temporal communities despite short term changes in instantaneous connectivities. For CLAN, the vertical bars where nodes have different colors represent a label reset, which are infrequent.



(a) Input Social Model (b) LPA

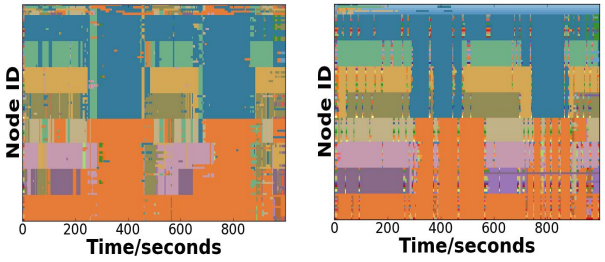
(c) Multislicing (d) CLAN

Fig. 5. All algorithms reveal similar evolution community structure. There are significantly more label resets in LPA and the instantaneous community structure detected by LPA does not always reflect the social model

In the second synthetic model, we have 100 nodes, including 5 travelers. Initially there are 8 communities, but they merge into 2 communities, then split into 8 communities and finally merge back into 2 communities. The results are shown in Figure 6. Once again, we demonstrate that CLAN is able to detect similar temporal communities to the centralized algorithm.

D. Variation of Information

The concept of *variation of information* [34] allows us to quantify the differences in clustering. Let X_N denote {nodes in x}, let $n(x, y)$ denote {nodes in x with label y} and let X_L



(a) Multislicing (b) CLAN

Fig. 6. Both algorithms reveal similar community structure, but CLAN is better able to separate the data mules from the other nodes as they eventually do not join communities

= {labels in x}. We define the intersection of two labels, i and j as:

$$n_{ij} = |n(A,i) \cap n(B,j)| \tag{6}$$

Variation of information of two clusterings, A and B is the sum of the distances between communities and is formally defined according to Equation 7. There is a special case, when $n_{ij} = 0$, and in this case the communities are disjoint over both clusterings and VI is defined to be 0.

$$VI(A,B) = \sum_{i \in A_L} \sum_{j \in B_L} -\frac{n_{ij}}{|A_N \cap B_N|} * \frac{log(n_{ij}^2)}{|A_N| * |B_N|} \tag{7}$$

Using this definition of VI, we can compare the results of the clustering algorithms relative to the ground truth social model used to generate the mobility. The average results over 10 runs of the 50 node synthetic network, described above, are shown in Figure 7. The spikes in VI with CLAN corresponds to community-wide label resets. The larger the community, the greater the spike in VI upon reset. There is some delay before CLAN and multislice can detect the merger at 550 seconds. This delay is due to the temporal aspect of these protocols, time is needed for link weights to increase before the merger can be detected. Generally, the communities detected by CLAN is closer to the ground truth than that of the other algorithms. A key difference is that the multislice algorithm experiences some instability during the first 150 seconds and CLAN does not, as reflected in Figure 5 and Figure 7.

E. Social Entropy

We use the notion of social entropy to control the ease with which a node joins a community. Nodes that frequently travel between communities have high social entropy making it more difficult for them to join communities. Since they do not join the community, they do not cause resets when they leave. With both synthetic traces described in Section IV-C, there are 5 travelers with no community attractions. They each belong to their own community and move independently between nodes selected at random. These nodes are shown by the 5 shades of blue at the top of Figure 5(a).

LPA and multislice are not able to isolate the travelers from the other nodes, as can be seen in Figure 5. With CLAN,

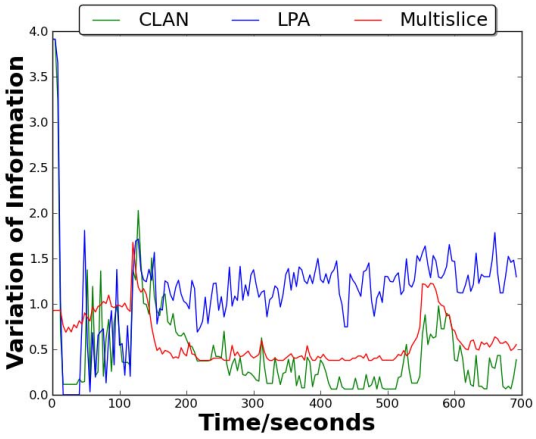


Fig. 7. On average, the community structure detected by CLAN is closer to that of the ground truth social model

however, the travelers eventually form their own individual communities. There is some learning time during which their social entropy grows. With high enough social entropy, the travelers do not join communities regardless of their position, hence the 5 shades of blue at the top of Figure 5(d) and Figure 6(b). These nodes do not trigger resets, thus there are few vertical, multicolored bars in Figure 5(d) and Figure 5(b).

F. Sensitivity

We vary CLAN's sensitivity parameter σ, as described in Section III-B and compare the results using variation of information. Figure 8 shows the average variation of information over 10 simulation runs with different seeds. It is clear that a sensitivity of $\sigma = 1$ or $\sigma = 2$ is insufficient, as the VI curve remained high; indicating that the split after 150 seconds was not completely detected. With a sensitivity of $\sigma = 1$, a node must lose all of its link before it starts a reset. This can only happen if a node becomes separated from every other node in the network. Without label resets, a dominant label takes hold and is propagated as nodes move about. As the value of sensitivity, σ, increases, the frequency of resets increases and the time to detect changes in communities decreases. Sensitivities of $\sigma = 3$ and $\sigma = 4$ both detected clusterings similar to the ground truth throughout the experiment (VI is 0), but $\delta = 3$ causes fewer resets (spikes in VI) and takes longer to detect changes (temporarily elevated VI after a change in ground truth). The value of σ allows a trade-off between time to detect change, and the overhead of resets.

V. Community-based Hierarchical Routing

Temporal community detection is of particular interest to the MANET community as it can be used as the basis for *scalable hierarchical routing*. The use of CLAN is more practical than other hierarchical approaches as communities do not need to be static or predefined, leaders do not have to be predesignated and communities are based on connectivity rather than geography.

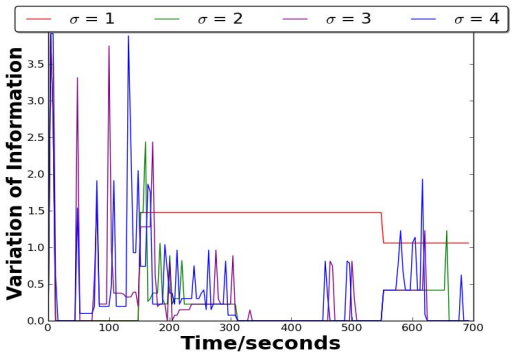


Fig. 8. In CLAN lower sensitivity σ results in fewer resets (spikes in VI), but it is more difficult to detect changes in community structure (elevated VI)

With a few modifications of OLSR, we instantiate community based routing using CLAN. Nodes advertise their labels in their periodic hello messages, allowing them to use CLAN algorithm to discover the evolving community structure of network. As in OLSR, each node transmits topology control (TC) messages containing a list of its current neighbors. Unlike OLSR, TC messages contain the source's label and are only retransmitted by members of the same community. Thus CLAN restricts TC messages to the community of the source unlike OLSR which floods TC messages throughout the network. This results less state at each node and reduces total overhead. The greater the number of communities, the more significant the reduction. Nodes use the TC messages to populate their intra-community routing table using Dijkstra's algorithms. Nodes will also be able to compute paths to nodes that have a link to the community but belong to an adjacent community. Thus nodes will be aware of adjacent communities and be able to compute the shortest path to each such community.

A node whose label is the same as its identifier is designated the community leader and these community leader form a level above the individual nodes in the hierarchy. Community leaders augment their TC messages to contain a list of community members and a list of adjacent community labels. These augmented TC messages are sent only upon convergence of CLAN and at a reduced frequency (one out of ten TC messages). The community leaders' augmented TC messages are propagated throughout the network. Nodes then use community leaders' adjacency to populate their inter-community forwarding table with the best adjacent community to every community in the network. Nodes also maintain a list, mapping each node to its last known community label.

When a node receives a packet, it looks up the community label of the destination and then use the intra-community or inter-community routing table as appropriate. When routing to an adjacent community, the node must have at least one member of that community in it's intra-community forwarding table and forwards the packets to the closest such node. Packets are routed from one community to another until it arrives at the destination community, where the intra-community forwarding table can be used.

We stop at this two-layer hierarchy, but for very large networks, the community leaders can be further clustered, using CLAN, to form additional levels in the hierarchy. We tested this routing protocol using the mobility pattern shown in Figure 6, with eight initial communities, which merge into 2 communities and then splits into eight communities again. The results are shown in Figure 9. We notice that with smaller sized networks, the difference in performance between the protocols are marginal. However, as the size of the network increases, the the CLAN incurs significantly less overhead while achieving comparable delivery and latency. With a 100-node network, CLAN incurs roughly half the overhead as OLSR. These results show that CLAN can leverage community structure to significantly reduce overhead and thereby improve scalability of routing protocols.

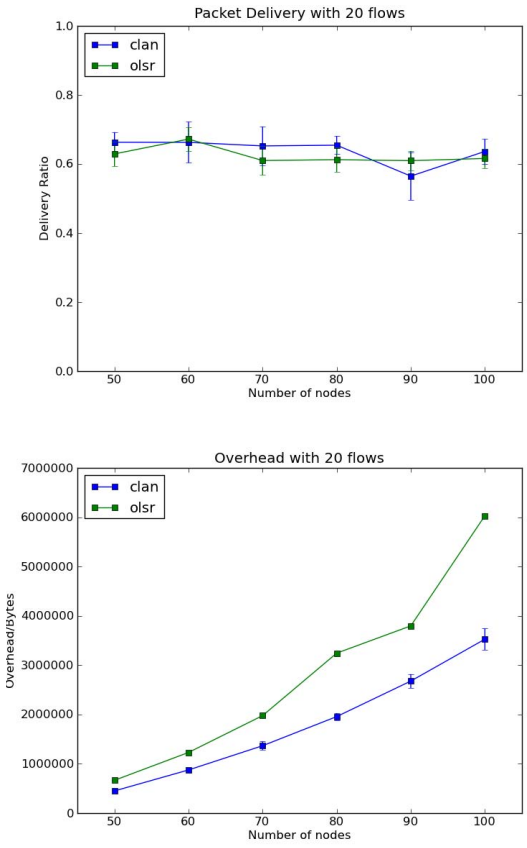


Fig. 9. The use of CLAN significantly reduces overhead without significant impact on delivery.

VI. Conclusions

We have proposed a distributed and efficient algorithm for detecting temporal communities in wireless networks. The algorithm does not need global topology information, as it utilizes 1-hop neighbor discovery messages. It also does not

need to discretize time into snapshots, rather it continuously updates the community structure efficiently as the network involves. The temporal community structure is further improved by assigning time-varying weights to links based on the uptown of the link and the variability in the set of neighbors of a node. Our distributed algorithm finds a temporal community structure as good as a centralized offline algorithm in our tests.

Some future work includes a more exhaustive study of the CLAN parameters, including more rigorous guidelines on how they can be set in a given environment. These include σ: the sensitivity for issuing resets, the minimum degree that a node is required to have to be able to resets, ψ: the time to detect convergence of LPA, and τ: the hello interval. Finally, we plan to explore further the effects that asymmetrical link weight updates using social entropy have in a variety of environments.

Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (Network Science CTA). Any opinions, findings and conclusions or recommendations are those of the author(s) and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Q. Li and J. J. Garcia-Luna-Aceves, "Opportunistic routing using prefix ordering and self-reported social groups," *Proc. IEEE International Conference on Computing, Networking and Communication*, 2013.

[2] A. Pietilänen and C. Diot, "Dissemination in opportunistic social networks: the role of temporal communities," *In Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2012.

[3] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2010.

[4] V. Kawadia, N. Riga, J. Opper, and D. Sampath, "Slinky: An adaptive protocol for content access in disruption-tolerant ad hoc networks," in *International Workshop on Tactical Mobile Ad Hoc Networking*, 2011.

[5] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structure in large-scale networks," *Physical Review E*, vol. 76, Sep. 2007.

[6] T. H. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," *RFC 3626; Available: http://www.ietf. org/rfc/rfc3626.txt*, 2003.

[7] T. Strayer, V. Kawadia, A. Caro, S. Nelson, D. Ryder, C. Clark, K. Sadeghi, B. Tedesco, and O. DeRosa, "Cascade: Content access system for the combat-agile distributed environment," in *IEEE MILCOM 2013*.

[8] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, 2010.

[9] P. Mucha, T. Richardson, K. Macon, M. Porter, and J. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *science*, vol. 328, no. 5980, 2010.

[10] V. Kawadia and S. Sreenivasan, "Sequential detection of temporal communities by estrangement confinement," *Scientific Reports 2*, 2012.

[11] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *In Proc. IEEE INFOCOM*, 2006.

[12] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003.

[13] M. Bakht, S. Nelson, N. Thompson, and R. Kravets, "Mercury: Leveraging clustering in opportunistic networks," in *Wireless Days (WD), 2011 IFIP*, 2011.

[14] Y. Chen, W. Zhao, M. Ammar, and E. Zegura, "Hybrid routing in clustered dtns with message ferrying," in *ACM MobiOpp*, New York, NY, USA, 2007.

[15] N. Sarafijanovic-Djukic, M. Piorkowski, and M. Grossglauser, "Island hopping: Efficient mobility-assisted forwarding in partitioned networks," in *IEEE SECON*, 2006.

[16] J. Coleman, *An introduction to Mathematical Sociology*. Collier-Macmillian, 1964.

[17] P. Jonsson, T. Cavanna, D. Zicha, and P. Bates, "Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis," *BMC Bioinformatics*, vol. 7:2, 2006.

[18] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69:026113, 2004.

[19] M. J. Barber and J. W. Clark, "Detecting network communities by propagating labels under constraints," *Phys. Rev. E*, vol. 80, no. 2, 2009.

[20] G. Tibély and J. Kertész, "On the equivalence of the label propagation method of community detection and a potts model approach," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 19–20, 2008.

[21] M. Rosvall and C. T. Bergstrom, "Mapping change in large networks," *PLoS ONE*, vol. 5, no. 1, 01 2010.

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

[23] P. Mucha, T. Richardson, K. Macon, M. Porter, and J. Onnela, "Community structure in time-dependent multiscale and multiplex networks," *Science*, vol. 328, 2010.

[24] H. Dang and H. Wu, "Clustering and cluster-based routing protocol for delay-tolerant mobile networks," *IEEE Transactions on Wireless Communications*, 2010.

[25] O. Drugan, T. Plagemann, and E. Munthe-Kaas, "Detecting communities in sparse manets," *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1434–1447, 2011.

[26] S. V. Dongen, "Graph clustering via a discrete uncoupling process," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 1, 2008.

[27] J. Reichardt and S. Bornholdt, "Statiscal mechanics of community detection," *Physical Review E*, vol. 74, no. 1, 2006.

[28] N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai, "Overlapping communities in dynamic networks: their detection and mobile applications," *MOBICOM*, 2011.

[29] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Stat Mech.: Theory and Experiment*, vol. 2008, no. 10, 2008.

[30] J. Ramming. Darpa baa05-42 proposer information pamphlet.

[31] D. Caprioni and A. Russo, "Small unit operations situation awareness system (suo-sas) radio architecture and system field testing results." *Military Communications Conference*, 2003.

[32] X. Lu, Y.-C. Chen, I. Leung, Z. Xiong, and P. Lio, "A novel mobility model from a heterogeneous military manet trace," *International conference on Ad-hoc, Mobile and Wireless Networks*, 2008.

[33] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 3, pp. 59–70, 2007.

[34] M. Meilă, "Comparing clusterings by the variation of information," *Learning Theory and Kernel Machines Lecture Notes in Computer Science*, vol. 2777, 2003.