MOPS: Providing Content-based Service in Disruption-tolerant Networks

Feng Li and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

***Abstract*—Content-based service, which dynamically routes and delivers events from sources to interested users, is extremely important to network services. However, existing content-based protocols for static networks will incur unaffordable maintenance costs if they are applied directly to the highly mobile environment that is featured in disruption-tolerant networks (DTNs). In this paper, we propose a unique publish/subscribe scheme that utilizes the long-term social network properties, which are observed in many DTNs, to facilitate content-based services in DTNs. We distributively construct communities based on the neighboring relationships from nodes' encounter histories. Nodes within a community directly communicate when events and interests match since they have strong intra-community relationships. Brokers are deployed to bridge the communities, and they adopt a locally weighted pub/sub scheme which combines the structural importance with subscription interests, to decide what events they should collect, store, and propagate. Different trade-offs for content-based service can be achieved by tuning the closeness threshold in community formation or by adjusting the broker-to-broker communication scheme. Extensive real-trace and synthetic-trace driven simulation results are presented to support the effectiveness of our scheme.**

***Index Terms*—Broker, community, disruption-tolerant networks (DTNs), forwarding process, localized algorithms, publish/subscribe, social network analysis.**

I. Introduction

Content-based service [1] is a novel style of communication that associates source and destination pairs based on the actual content and interests, rather than by letting source nodes specify the destination. Content-based networking allows ad hoc and autonomous access to content. The decoupling of information producers and receivers allows for greater scalability and a more dynamic network topology, which makes content-based service suitable for many possible network applications. In this paper, we adopt the *publish/subscribe* (pub/sub for short) scheme, which is an asynchronous messaging paradigm, to provide the content-based service. The *subscribers*, which are the information consumers, express their interest in certain events without knowledge of what publishers there may be. The *publishers*, which are the information producers, issue newly detected events without having to specify the receiver. The *brokers*, which match events with interests, are deployed as the interface between the publisher and subscriber.

[0]This work was supported in part by CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240. Email: jie@cse.fau.edu

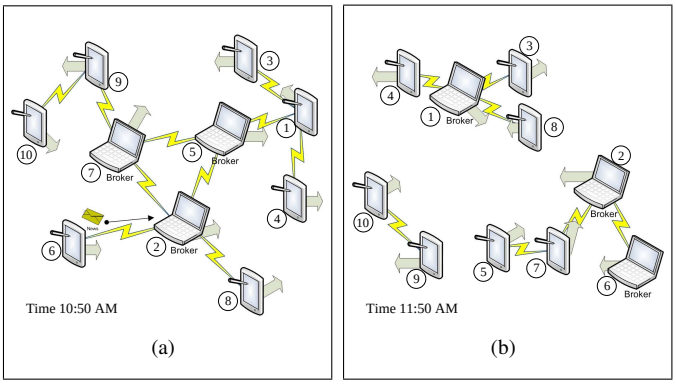


Fig. 1. Example pub/sub services in the DTN.

In highly mobile environments, such as disruption-tolerant networks (DTNs) [2], the network topology constantly changes and end-to-end paths can hardly be sustained. On one hand, pub/sub schemes can provide a high degree of flexibility and adaptability to change by decoupling the source and destination, which can greatly facilitate information dissemination in the DTN. Guided by the interests, events are replicated and propagated. No end-to-end path is needed in the pub/sub scheme. On the other hand, with the increasing popularity of mobile handheld devices, which have ad hoc wireless communication abilities (e.g. Bluetooth), the DTN also became more and more attractive. There is a pressing need to extend pub/sub to DTNs. An example pub/sub service where students can share campus news videos over the network composed by their mobile devices is shown in Fig. 1. In Fig. 1, nodes labeled "brokers" can be publishers or subscribers, and other nodes are publishers and subscribers.

Most existing pub/sub research [3], [4], [5] concentrates on fixed networks or networks with very limited mobility. In these schemes, the brokers are deployed according to connectivity metrics and the topology is dynamically maintained if needed. However, when the networks are highly dynamic, maintenance costs will be higher than acceptable, thus rendering these schemes inapplicable to the DTN. As illustrated in Figs. 1(a) and (b), the connectivity can dramatically change and the relationships among publishers, subscribers, and brokers, as determined by traditional connectivity metrics, break.

Based on multiple sets of real DTN traces, such as Haggle [6] and Reality Mining [7], we observed that a long-term

closeness metric can be abstracted to depict the neighboring relationship between nodes. Based on this inherent property of the DTN, we propose a Mobile cOmmunity-based Pub/Sub scheme (MOPS), to facilitate the content-based service.

In the pub/sub scheme, two extremes can be adopted. One is the pure *push* strategy, which can achieve a short latency and high delivery ratio at the cost of high network traffic. The other is the direct *pull* strategy, which reduces the total number of event forwardings at the cost of large latency and small delivery ratio. Since applications in DTNs usually operate best under a moderate trade-off between latency, delivery ratio, and redundancy, a combination of both push and pull strategies is needed in the design. Therefore, the challenges of the MOPS include designing the pub/sub protocol which combines push and pull, determining the interface known as the push-pull boundary, and deploying brokers to bridge the boundary.

We first propose a community construction scheme to determine the push-pull boundary. Nodes in the DTN distributively construct clique-style communities based on limited hops of local information. These communities have desirable properties of controllable diameter and strong intra-community connections which can eventually facilitate the pub/sub service.

Within the community, nodes broadcast interests and publishers send events directly to the nodes that expressed interest in the category of the event. The brokers are deployed on the boundaries of communities to bridge the events and interests inside and outside the community. The internal interests of a community are aggregated by each broker in that community, and propagated to the one-hop neighboring communities.

When a broker meets another node, it collects events from the node according to the order of the aggregated interests, which is decided by the weight of the interests' category. In MOPS, the weight is uniquely designed so that it combines the brokers' structural importance together with the aggregated subscription interests. By structural importance, we mean the importance of node position in network topology. Moreover, by setting different default weights for the categories without current subscriptions, the brokers' push or pull strategy can be controlled and the performance trade-off can be further tuned.

The contributions of this paper are summarized as follows:

1) We present a closeness metric to depict the relationship between each pair of nodes in the DTN. It captures the core of temporal and spacial encounter information.
2) We design a local community based pub/sub scheme. A closeness threshold can be turned to adjust the property of the local communities. The pub/sub service utilizes the efficient intra-community communication. Brokers are deployed on the boundaries of the communities.
3) We propose a weighted pub/sub scheme to improve the communication efficiency. The brokers aggregate interests and efficiently collect the events according to the weight that combines the brokers' structural importance together with the aggregated subscription interests.
4) We verify the effectiveness of our schemes based on real mobile traces and simulations.

II. Preliminary

We discuss the basic DTN model used in this paper, link abstraction, and general pub/sub process in this section.

A. Network model

DTNs attempt to route packets via intermittently connected nodes. Many store-and-forward style schemes [8], [9], [10], [11] are proposed to deliver packets in this challenging environment. In existing DTNs, such as UMass DieselNet [12] and MIT Reality Mining [7], real objects' movements usually follow patterns that are repetitive to a certain extent. In this paper, we consider the DTN where nodes are constantly moving and the topology is highly dynamic. However, we also assume the nodes' movements are random but statistically repetitive to a certain extent. Therefore, the encounter history can be used to predict the future contacts.

Moreover, we focus on data-centric communication rather than IP- or other-identity-based. Users want to obtain content of particular categories such as music, news, or photos on their communication devices. Similar network settings are also studied in recent works [13], [14], [15], but with a different focus. Two nodes can exchange information when moving into each others' communication ranges. We also assume the contact duration is limited due to nodes' movement, and it is possible that only partial content can be exchanged before two encountered nodes depart from each other.

Some recent studies [16], [17], [18], [19], [20], [21] based on real mobile traces reveal that DTNs show certain social network properties. For example, two important metrics, familiarity and centrality, are measured based on nodes' direct or indirect observed encounters and used to guide the forwarding in [17]. In this paper, a closeness metric is also abstracted from the encounter history to represent the long-term relationship between each pair of nodes.

B. Relationship abstraction

In DTNs, each node can record the encounter time and duration whenever it meets another node. However, nodes' original knowledge, which includes both temporal and spacial information, is abstracted into a single *closeness* metric $c_{uv} \in [0, 1]$ for nodes u and v. The closeness metric abstracts the time-space relationship and indicates the prediction towards the forwarding opportunities between nodes. Larger closeness c_{uv} indicates a better future contact opportunity.

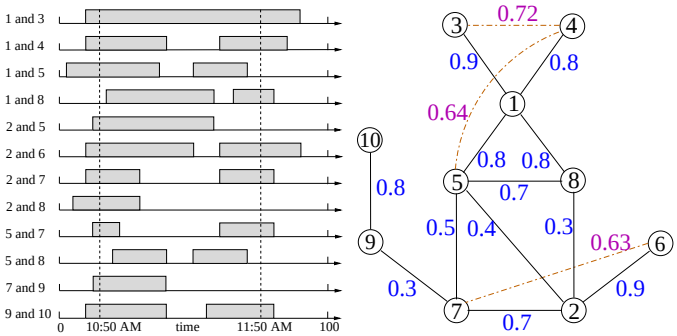
To measure c_{uv}, a training time window should be adopted, and c_{uv} is determined by u and v's encounter history in this training time window. The average separation period, which is represented as $AVG(D_{uv})$, is a comprehensive metric to start the time-space abstraction since it reflects both the frequency and length of the encounters. Smaller $AVG(D_{uv})$ indicates shorter communication latency between u and v. We apply the Gaussian similarity function [22] to normalize $AVG(D_{uv})$ as follows and denote the resulting metric as closeness c_{uv}:

$$c_{uv} = \exp(-\frac{(AVG(D_{uv}))^2}{2\sigma^2}). \quad (1)$$

Here, σ is a scaling parameter [22] for the separation period.

We model the neighboring graph of a DTN as G, where each vertex u in G corresponds to a node in the DTN; each edge $< u, v >$ in G represents that two nodes have encountered, and it is associated with a closeness metric c_{uv}. We use d_u to denote the degree of node i, which is the sum of the closeness of all edges connecting i.

An example is shown in Fig. 2. Fig. 2(a) illustrates the encounter history of the DTN in Fig. 1, and the transient scenarios in Fig. 1 correspondent to two dotted lines. The links in Fig. 2(b) are derived from Fig. 2(a). The length of the training time window is 100 and $\sigma = 25$. Since node 5 and 8 met twice at 20 and 50 and the durations are both 20, we can derive that $c_{58} = 0.7$ as shown in Fig. 2(b).



(a) Encounter history (b) Neighboring graph

Fig. 2. An example neighboring graph abstracted from the encounter history of the DTN in Fig 1.

C. Pub/Sub process

Each node in a DTN may take the responsibility of publisher, subscriber, or broker. In MOPS, each data unit is marked with some metadata describing its content. For the sake of simplicity, we assume that each event report X_i is marked with one single category X, describing its content. The set of possible categories is assumed to be finite. Since pure push and pull can only produce extreme performance, we define the scope called the push/pull boundary and only allow subscribers(publishers) to directly pull(push) in the boundary.

1) Subscription: In MOPS, a subscriber broadcasts its subscriptions to other nodes in its pull boundary. The interests may change over time. A node u's interest list includes $\{(X, w)\}_t$. Here X represents a specific category of interest, and the weight w is in the range $(0, 1]$ expressing the level of interest; t is the list's generation time.

2) Publication: Having detected new events, nodes generate event reports and publish them to their associated push boundary. Each event report includes (X_i, t, TTL), where t is the content generation time and TTL is the event report's time-to-live (TTL).

Each message will include the node ID of the source, the destination, and the next hop when these messages are wrapped for forwarding. Each node has an event buffer. When the TTL has expired, the event will be removed from the buffer. We are not assuming any constraint on the nodes' buffer since buffer management is not the focus of this paper. Each publisher has a pub list which contains the forwarding record of each event.

Brokers are deployed to expand the push and pull boundary. Brokers aggregate the interests inside the boundary and collect events from outside the boundary to satisfy the interests. They also help to further propagate the events published inside the boundary to the outside.

All communications are based on the gossip style data exchange rules for each encounter duration in the DTN: when node u meets node v, u sends a query message containing its interest list. Node v checks the availability of such content in its event buffer and transmits the new events which match the received interest, in decreasing order of weight. v follows the same process during the encounter.

To accommodate the above pub/sub process in challenging DTN environments, several points need to be further discussed. First is the definition of local association which determines the push/pull boundary. Since the network topology is transient, the locality should have a new definition. Second is broker assignment and communication. Broker nodes should work in a way that efficiently utilizes the locality to enhance the pub/sub performance.

III. General MOPS Scheme

In MOPS, clique-style communities are constructed to determine the local association. Different intra- and inter-community pub/sub schemes are then applied on top of the communities.

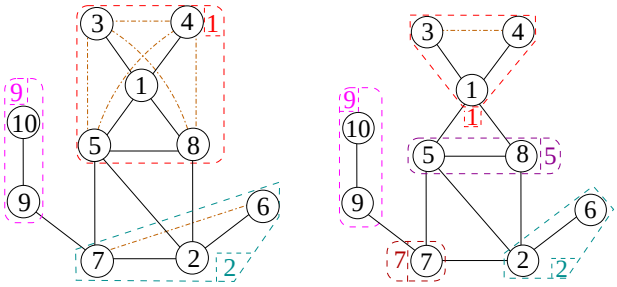
A. Local community

The community is a reflection of locality. As a criterion to determine whether the relationship between two nodes is strong enough to claim they are local neighbors, we adopt a threshold T on the closeness metric c associated with each link in the neighboring graph. For two nodes u and v in the DTN, if there is an edge $< u, v >$ in G and $c_{uv} > T$, we consider u and v to be *local neighbors*. Based on Equation (1), the expected delay between local neighbors u and v has the upper-bound $-\sqrt{2 \cdot \ln T} \cdot \sigma$ with this threshold-based filtering.

In graph theory [23], a *clique* is a subgraph in which every vertex is connected to every other vertex in the graph. We extend the idea and define *local community* in the DTN as follows:

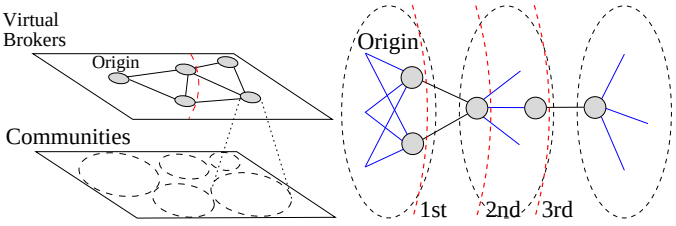
Definition 1: Local community: For any pair of nodes in the community, they are local neighbors to each other, i.e. a link exists such that the closeness c of the link is larger than the threshold value T.

However, the above definition is too restrictive. Two nodes without a direct relationship may have a common neighbor that has a close relationship with both nodes. From the angle of data forwarding in the pub/sub, we should also consider such a multi-hop neighboring relationship. Therefore, we extend the idea of local neighbors and define a virtual link as follows:



(a) T=0.5 (b) T=0.7

Fig. 3. Threshold T decides local community property. Dashed boxes represents community, and numbers in small squares are the community IDs.



(a) Broker mapping (b) Three scenarios

Fig. 4. Inter-community pub/sub.

Definition 2: Virtual link: If at least one path up to k-hops between nodes u and v exists, a virtual link can be used to represent u and v's neighboring relationship. The virtual link will be associated with $c_{uv} = \max_{p \in P}\{\prod_{<i,j>\in p} c_{ij}\}$, where P represents the set of all paths between u and v which are less than or equal to k-hops.

Here the *path closeness* for path p is $\prod_{<i,j>\in p} c_{ij}$, which is a product of all the edge closenesses along the path. The path closeness indicates the expected delay along that path. Based on Equation (1), the path closeness naturally reflects the upper-bound of the combined delay on edges $< i, j > \in p$.

We use the maximum value to represent the virtual link closeness, as opposed to the sum of all paths, since c_{uv} should reflect the shortest expected delay between nodes u and v. Similar to direct links, only when the closeness on the virtual link $c_{uv} \geq T$ will u and v be considered local neighbors to each other. For a pair of local neighbors, the path with the largest closeness between them is denoted as a *local connection*. A community construction scheme which only utilizes the local information is presented in the Appendix.

The distribution of the closeness metric in real DTNs indicating the closeness of a virtual link is usually decided by a path with a very small number of hops. As case studies, we analyzed data from the Haggle project dataset [6] and MIT Reality mining [7]. It is clearly indicated that closeness between two nodes is mainly decided by a direct link (60.8% in Reality and 12.2% in Haggle) or a 2-hop link (30.9% in Reality and 83.3% in Haggle). Therefore, we can have a restriction of k (2 or 3) and only consider the paths up to k-hops when calculating the virtual link closeness. Fig. 2 contains several example virtual links, such as the one between nodes 3 and 4. The 2-hop virtual link is associated with $c_{34} = 0.72$.

By adjusting T, the community shows different properties that we can use to achieve desirable trade-offs. Figs. 3(a) and (b) illustrate the results of our distributed algorithm with $k = 2$ when $T = 0.5$ and $T = 0.7$. With higher T, the total number of communities clearly increases and the internal links are stronger. Therefore, T is the first controllable parameter that MOPS can utilize to achieve desirable trade-offs.

B. Intra-community pub/sub

Each node, with the role of the subscriber or broker, will first prepare its own interest list. The subscriber or broker will send its interest list to every other node in the community through the local connection, i.e. the interest list is broadcast throughout the community.

When a publisher u detects an event, it checks the stored interest lists of other nodes in the same community. u should send the event report to all matching subscribers/brokers through the local connection.

If the local connection is a direct link, the data will be exchanged directly. Otherwise, when the sender meets the first node on the local connection of a virtual link, it will mark its data with the intended destination and the data will be propagated towards the destined local neighbor.

Fig. 5(a) shows an example of the intra-community pub-sub process for the DTN shown in Fig. 3(a). Subscriber 1 broadcasts its interest list $\{(A, 1), (C, 1)\}_t$ in community 1. When publisher 3 detects event C_1, it will send C_1 to node 1 according to the stored interest list.

Since each pair of nodes in a community has a strong connection between them, the delivery rate will be high and the latency will be small using the above scheme when publisher and subscriber reside in the same community. However, the publisher and subscriber of an event can reside in different communities. Brokers are needed in the event forwarding process in this situation.

C. Inter-community pub/sub

We assign the duty of inter-community communication to the brokers. The broker is responsible for matching and disseminating internal and external events and interests.

The definition of local community guarantees the strong connection within community. The communities can now be regarded as the units to conduct the pub/sub. Therefore, each community can be regarded as one *virtual broker* that handles subscription aggregation, event collection, and propagation, as shown in Fig. 4(a). In the MOPS scheme, the internal subscription interest of community A will be aggregated and propagated to its one-hop neighboring communities. The brokers in A's direct neighboring communities will help to collect the events and send them back to A. The interest is not further propagated for two main reasons. First, each node's pub/sub scope is largely expanded and the community level

two-hop path is long enough to collect most of the interested events in most DTN settings. Second, the complexity of the scheme increases to an impractical level since loops need to be avoided when the interest is further propagated.

The virtual broker's responsibility will be distributively shared by a select group of *gateways*. Gateways are nodes that have direct neighboring relationships with nodes in other communities. There are three possible scenarios as illustrated in Fig. 4(b) of gateway communication, which need to be considered when we distributively implement the virtual broker.

Each gateway should summarize the interest lists of the nodes in its own community into an internal interest list, then collect the matching events from outside of the community. This intra-community non-gateway to gateway communication is considered to be the first scenario in Fig. 4(b). An example is broker 5 in Fig. 5(b).

The second scenario in Fig. 4(b) is the direct communication between gateways from different neighboring communities. Each gateway will summarize the interests of other communities and form the external interest list set. The gateway will then represent the external community to collect events from its own or other neighboring communities. The communication between gateways 2 and 5 in Fig. 6(a) belongs to this scenario. Gateway 2 needs to aggregate the interest list from community 1 in its external interest list. Then gateway 2 can collect events in community 2 according to its external interest list.

Moreover, a gateway u may propagate its external interest lists to (and request help from) another gateway v in its own community but connects to other neighboring communities that are unreachable from u. This is considered to be the third scenario. Gateway v needs to setup a relay interest list set. The difference between an external and a relay interest list is that the relay interest list won't be considered when the gateway collects events from its own community. Gateway 2 will request 7 to collect events for community 1 in Fig. 6(b). Gateway 7 considers gateway 2's external interest list of community 1 as the relay interest list.

Since gateways are the physical nodes that distributively implemented the pub/sub interface among communities, We use the term broker in the subsequent discussion to refer to gateway. An efficient broker communication mechanism can greatly improve the performance of the pub/sub scheme in DTNs. Therefore, we present a unique weighted scheme with more detail in the next section.

IV. Weighted Broker Communication

In the MOPS scheme, the brokers are guided by a unique weighted scheme. The scheme includes four components: weight calculation, interest propagation, default weight determination, and broker pruning.

A. Weight calculation

Since the contact duration is usually limited, we need to differentiate the importance of the subscription interest, which is reflected by the weight in MOPS. There are several rules that we can derive before we design the weight. First, the internal interests should have higher priority than external interests when a broker collects events from other communities. Second, an event category should have higher priority when more nodes or communities are requesting the category simultaneously. Third, the weight should decay and reflect the closeness between the broker and the subscriber. The broker should put more strength towards collecting events for nodes that are closer to it. Before the interest propagation starts, each node sets weight $w = 1$ for all the categories in its own subscription list. We consider the three scenarios in Fig. 4(b).

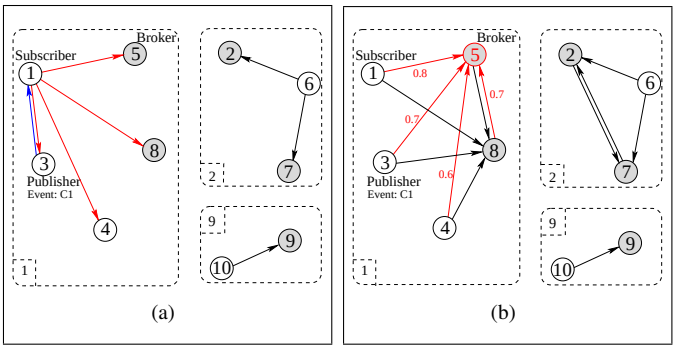


Fig. 5. Intra-community pub/sub examples. Shadowed nodes are the broker.

Internal weight calculation. Brokers should aggregate the interests from their own community and assign weight I to each category X in the first scenario in Fig. 4(b). For a broker u, the weight of a category X in the internal interest list of broker u is the sum of u's closeness to all the subscribers of X in the same community. Each broker u will form its own internal interest list. More formally, the weight I of a category X in u's internal interest list should be calculated as:

$$I_u = \sum_v c_{uv}. \tag{2}$$

Here, v is in the same community as u, and v expressed interest in X.

Take broker 5 in Fig. 5(b) as the example. It aggregates the interest lists of nodes 1, 3, 4, 8, and 5 to generate the internal interest list of community 1. Assuming that event category C is included in both 1 and 8's interest list, 5 will set I_5 for C as $1 \cdot 0.8 + 1 \cdot 0.7 = 1.5$.

External weight calculation. For the second scenario, a broker u's external interest list set should include the internal interest lists from any broker v in u's neighboring community. A broker u may have a neighboring relationship with multiple brokers in u's neighboring community A. The maximum decayed weight is used to characterize the category since it represents broker u's best possible contribution to the propagation of events in this category. The summarized weight E of each category X in its external interest list should be calculated as follows:

$$E_u = \sum_A \max_{v \in A} \{c_{uv} \cdot I_v\}. \tag{3}$$

Here, A is the distinct community and v is a broker of A in u's collected external interest list set. In other words, for each

community A in u's collected external interest list set, one broker v in with largest $c_{uv} \cdot I_v$ for a category X is selected.

As shown in Fig. 6(a), broker 2 will construct its external interest list based on both broker 5's and broker 8's internal interest list of community 1. Taking E_2 for event category C as the example, $E_2 = \max\{0.5 \cdot 1.5, 0.4 \cdot (1 + 0.8)\} = 0.75$.

Relay weight calculation. In the third scenario, a broker v may forward an aggregated external interest list for each neighboring community A to broker u. As an example, broker 2 in Fig. 6(b) forwards its external interest list of community 1 to broker 7. The broker u will store the lists in the relay interest list set and calculate the weight $R_u(A)$ of the relay interests $R_u(A) = \max\{c_{uv} \cdot E_v(A)\}$ for each community A.

Broker u then combines its external interest lists set with the relay interest lists set. For each community A in the expanded set, a maximum decayed weight $\max\{E_u(A), R_u(A)\}$ will be found. ER_u is the sum of this weight for all neighboring communities. ER_u is used when broker u is collecting events from another neighboring community.

$$ER_u = \sum_A \max\{E_u(A), R_u(A)\}\}. \tag{4}$$

Here, broker $u \in$ community B and A is the distinct neighboring community in u's collected external interest list set.

The example can be found in Fig. 6(b). Broker 7 receives a relayed interest list from 2 and constructs its external interest list based on the lists from 8 and 9. For event C, assume $I_9 = 1.7$, the $ER_7 = 0.3 \cdot 1.7 + \max\{0.8 \cdot 0.75, 0.3 \cdot 1.8\} = 1.11$.

To subscribe to the events produced in its own community, the broker will broadcast an interest list combining its self and external interests, with the weight for each category X calculated as:

$$w = \alpha \cdot S + \beta \cdot E. \tag{5}$$

Here, $S = 1$ if category X is in the broker's self interest list; otherwise $S = 0$; $\alpha + \beta = 1$ and $\alpha >> \beta$.

To subscribe the events produced in other communities, the broker will send an interest list combining its internal and external interests to each broker it encounters that belongs to another community, with w calculated as:

$$w = \alpha \cdot I + \beta \cdot ER. \tag{6}$$

The event exchange between two brokers from different communities is guided by the weight. Events in the category with higher weight will be exchanged earlier, and events with low weights may not get a chance to be transmitted.

B. Interest propagation.

When a broker u meets another node v, they will exchange interest lists and events based on their role and guided by the calculated weight. Before the communication begins, the broker u will prepare a self interest list and assign the same *default weight* to all the categories in its external interest list. When u encounters another node v, u decides its interests propagation steps based on the type of v. Therefore, if v is a:

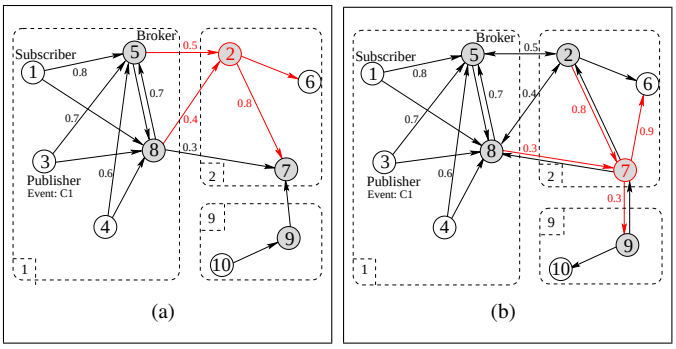


Fig. 6. Inter-community pub/sub examples.

- **Non-broker node from the same community**, u sets $w = \alpha \cdot S + \beta \cdot E$, and sends the interest list. After receiving the self interest list from v, u aggregates it into u's internal interest list.
- **Broker from the same community**, u sets $w = \alpha \cdot S + \beta \cdot E$, and sends the interest list. u also aggregates its external interest list set and sends a relay interest list for each of its direct neighboring communities. Broker u also adds or updates received relay interest lists from v.
- **Broker from other communities**, u sets $w = \alpha \cdot S + \beta \cdot ER$, and sends the interest list. u also sends its internal interest list to v. u will add or update received internal interest list of v into its external interest lists set.

After the interest list has been exchanged, u will send new event reports in the buffer to v in an order decided by the weight of each category. Events in the category that feature a larger w will be sent earlier. Only events with non-zero weight in v's interest list will be sent.

C. Default weight

For the events in the categories that are not currently subscribed by any nodes in the brokers' own and neighboring communities, the brokers can select to ignore these events, or adaptively collect them after the events with subscription have been exchanged. In MOPS, this selection is realized by setting the default weight of the unsubscribed categories.

If the default weight is a non-zero value, the broker will choose to receive the events in the unsubscribed categories when the meeting duration is long enough. These events will be stored for future subscriptions or being collected by the brokers from the communities that have external interest towards them. The default weight should be a small value such that it will not disrupt the order of subscribed events. The advantage of the non-zero default weight is that it would be more adaptable to interest change in nodes' subscription. It also helps in the rare situation where brokers in the subscriber's own and one-hop neighboring communities cannot reach the event's publisher. However, it also induces unnecessary event forwarding.

Another possibility is to set the default weight to zero. The pros and cons of this scheme are opposite to the above. The broker will only collect events that are subscribed by nodes in

TABLE I
Characteristics of three datasets

Dataset	*Haggle*	*Reality*	*Synthetic*
Device	iMotes	Phone	N/A
Network type	Bluetooth	Bluetooth	N/A
Duration (days)	3	246	10
Number of nodes	41	97	200
Number of contacts	22,459	54,667	Vary

its own or one-hop neighboring communities. The unnecessary event forwarding is reduced. However, the spare forwarding opportunity will be wasted after the subscribed events have been transmitted and the interest change in the subscriptions need a long time to be satisfied.

D. Broker pruning

Not all gateways are needed in the inter-community forwarding since this may create unnecessary redundancy. In MOPS, the actual brokers are selected from gateways using a pre-pruning scheme.

Multiple links in the neighboring graph between a pair of communities A and B may exist. However, to achieve a desirable trade-off, some of the links can be trimmed to avoid redundant transfers of the same event report between the communities for excessive number of times. A gateway should mark itself as the broker if and only if at least one of the inter-community links it connects to is not pruned.

Each gateway calculates its *betweenness* centrality for the community pair A and B it connects to. The betweenness centrality is a measurement of the structural importance of a node [24]. $b_{\bar{u}}$, which represents the combined forwarding capability of the gateways with higher betweeness centrality than u, is also calculated. If the combined forwarding capability satisfies the delivery requirement, u will be able to prune itself to reduce the redundancy without violating the desirable trade-off. More specifically, if both of the following conditions are satisfied, gateway u prunes itself from being the broker for the community pair A and B:

- u's centrality b_u is not the highest among the gateways;
- For other gateways with higher centrality, the combined similarity $b_{\bar{u}}$ is equal to or larger than a threshold B.

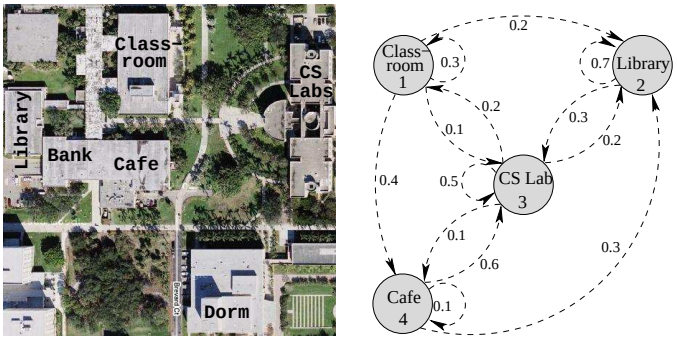
After the brokers are selected from the gateways, they can start to act as the bridges between communities. The details of the betweenness calculation are given in the Appendix.

V. Simulation Analysis

We have conducted simulations to evaluate the effectiveness of the MOPs scheme with both real and synthetic DTN traces.

A. Simulation setup

In our simulations, we compare the effectiveness of our scheme with three other techniques: pure push, direct pull, and k-nearest-neighbors [15] (Neighbors for short). In pure push, a node replicates an event it stores to every node it encounters that has not received a copy. In direct pull, a node only collects events that it has interest in from its directly-encountered nodes. Nodes in the Neighbors scheme aggregate the interests from the k nearest neighbors, and associate weight according to the popularity among the k nearest neighbors to each category of interest. Nodes then collect events according to the weights.



(a) FAU map (b) CS graduate

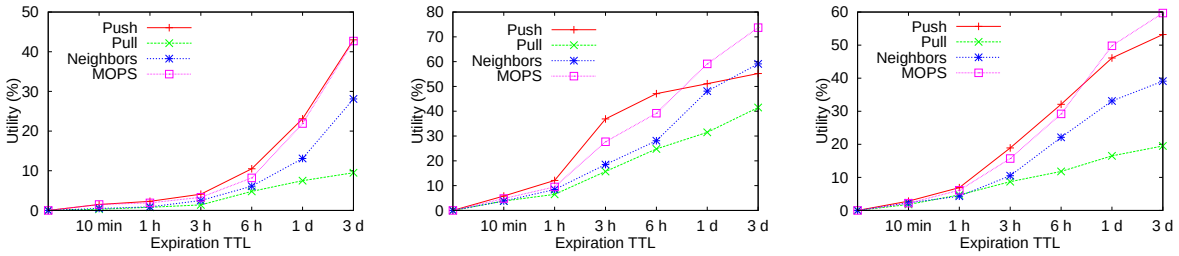
Fig. 7. The synthetic mobility traces are generated from map of FAU.

We primarily focused on two parameters: 1) *Utility:* for each event, the utility is reflected in the proportion of nodes interested in the event's category that receives the event; and 2) *Efficiency:* for each event, efficiency is defined as the number of interested nodes that receives the event to the total number of nodes infected by the event. We also investigate the *latency* and the cost in terms of *total number of forwards*.

DTN environment. We ran trace-driven simulations with two different datasets: Haggle project [6] and MIT Reality Mining [7]. In Haggle project, 41 iMotes were distributed to students attending Infocom 2005. In Reality, 97 smart phones were deployed to students and staff at MIT. In both datasets, bluetooth contacts were logged and provided. Each contact record includes the start time, end time, and ID of the nodes in contact. For each round of simulation, a portion (default 40%) of the dataset is used as the contact history. The remaining portion is used to evaluate the performance of pub/sub after the community detection and gateway pruning.

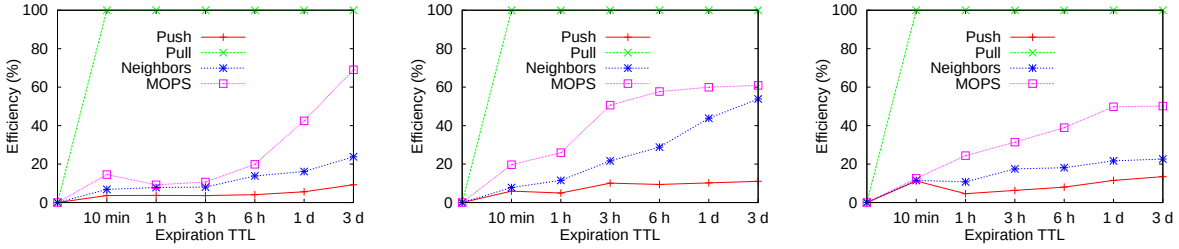
We generated synthetic traces according to a community mobility model proposed in [18], which is considered to be more realistic than *i.i.d.* models. The traces were generated using maps of the Florida Atlantic University (FAU) buildings as shown in Fig. 7(a). The class schedules and enrollment information of 200 graduate and undergraduate students from four departments were collected. The trace of a node, which represents a network device carried by a student, was generated according to a Markov chain as illustrated in Fig. 7(b). The states and probabilities were determined by the students' class schedules and enrollment information. If two nodes were in the same building at the same time, they had a probability (default 0.8) of contacting each other. The contact length follows a power-law distribution. We also assume there is no constraint on the nodes' event buffer.

Events and interests. We simulated the scenario of campus video news sharing. We assumed each event is 5 Mb of size and that the contact data rate is 500kb/s (Bluetooth). For each



(a) Reality (b) Haggle (c) Synthetic

Fig. 8. Performance comparison on utility.



(a) Reality (b) Haggle (c) Synthetic

Fig. 9. Performance comparison on efficiency.

simulation, nodes were uniformly selected to be the publisher of an event. Events were generated by a Poisson process, where the rate parameter λ is adjustable in the simulation.

The subscription interests and event categories were simulated by the the Number Interval model in [25]. In specific, an event is randomly associated with an integer value within the interval [1, 100] after being generated. Each category of events is represented by a random but distinct range within [1, 100]. For each category, a subscriber may have interest in it with a uniform probability p (default 0.2). An event matches a subscription only when the event's value falls into the subscription categories's range.

All packets had an expiration TTL, which represents the delay requirement. Each node knew only the contact history of itself before the community detection. Each simulation was repeated 30 times with different random seeds for statistical confidence. At each round, default $\alpha = 0.99$, $k = 2$, $B = 0.8$, $\lambda = 10$ events/hour, and $TTL = 1$ day. We adopted the average separation period of all nodes as σ.

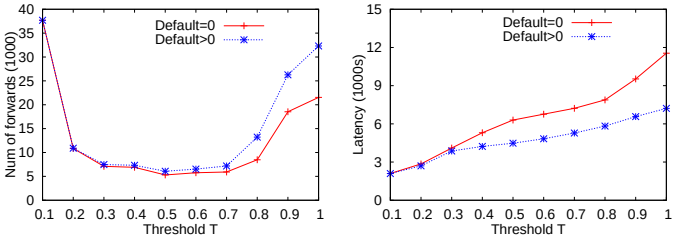
B. Simulation results

In the first experiment, we set the parameters $T = 0.5$ and default value $= 10^{-5}$, and compare MOPS with three other schemes. As shown in Figs. 8 and 9, the utility and the efficiency both increase as the delay requirement on the packet lessens. The three datasets represent three different DTN scenarios.

The Reality dataset is a scenario that contains many communities and the frequency of contacts is also lower than in the other two cases. All four schemes can only achieve a utility of less than 40% when the expiration TTL is three days. As expected, in this low contact rate environment, the most aggressive method, which is the pure push, produces the highest utility. The utility in the direct pull strategy is very low because the subscribers have only a very slight chance to meet the publisher directly in this environment.

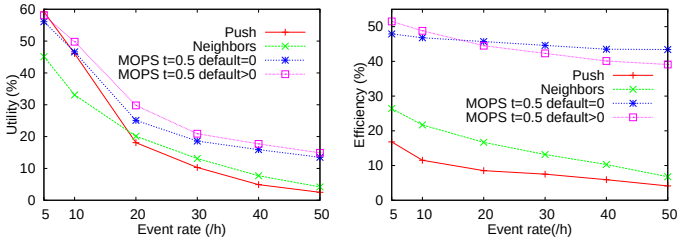
The MOPS scheme achieves a utility close to the upper bound indicated by the curve for pure pull, and clearly outperforms Neighbors in this case. The reason for this is that a packet will be broadcast at the community level in MOPS if the source and destination are in different communities in this low contact rate environment. Therefore, for the case where the packet forwarding is within a community, the source and destination may meet more than once; otherwise, the community level broadcast has a good chance of including the path with the shortest delay. The replication strategy in Neighbors which only depends on 2-hop paths is too conservative in this case, therefore it produces a lower utility as illustrated in Fig. 8(a).

The Haggle dataset contains fewer nodes and fewer communities. Nodes meet more frequently in this dataset, therefore, the utility of all four schemes is also higher in Fig. 8(b) than in the other two datasets. When the TTL is small, the pure push strategy still outperforms the other three schemes. However, when the TTL is larger, the utility does not increase much for the pure push because the nodes may buffer many events and unnecessary propagations waste the bandwidth in each exchange. The MOPS scheme shows a clear superiority over the other schemes. The weight guided inter-community communication in MOPS can relay events for a long path if necessary, while direct pull restricts to 1-hop and Neighbors restricts to 2-hops. The utility for the MOPS is also higher in this case than in Fig. 8(a) and (c). The reason for this



(a) Number of forwards (b) Latency

Fig. 10. Performance comparison by adjusting T.



(a) Utility (b) Efficient

Fig. 11. Strategy comparison with different event generation rate.

is because the communication involving brokers has a higher success rate in this scenario.

For the synthetic dataset, we observe contact frequency between that of the Haggle and Reality datasets. In Fig. 8(c), the MOPS scheme still shows a clear improvement in utility towards other schemes except pure push. When the events' TTL is long, MOPS is even better than pure push because the contact bandwidth is efficiently utilized as the brokers will first relay events that they can contribute more, and the weight helps brokers to avoid blind event propagation. MOPS also shows the largest margin of utility towards the Neighbors scheme in this scenario because the difference in nodes' centrality is larger. The Neighbors scheme is biased towards nodes with smaller centrality. Subscribers with low centrality may not receive the corresponding events since they may not be the k-nearest neighbor for any node. These nodes have a low chance of being infected even when the TTL is long enough in this scenario.

In all three scenarios in Fig. 9, the efficiency of the MOPS scheme is higher than that of Neighbors and pure push. In the MOPS scheme, only the necessary brokers would be infected among those nodes which are not the subscriber. Intra-community communications are directly conducted between subscribe and publisher. The brokers will be infected to satisfy the subscription interests only when the subscriber and publisher reside in different communities. Moreover, the weight which combines the structural importance and subscription interests makes the brokers differentiate events, which further increases efficiency. The efficiency of the Neighbors scheme is lower than MOPS because the neighbors of a subscriber with high centrality get infected repeatedly and the subscriber with low centrality may not have neighbors to help. Pure push shows very poor efficiency in all cases due to the completely blind diffusion. The efficiency of the direct pull strategy is always 1 since there is no relay in this scheme.

Figs. 10(a) and (b) illustrate the effect of adjusting the threshold T using the synthetic trace with $TTL = 1$ day. We observe that a moderate T of about 0.5 in this scenario makes the MOPS scheme produce a moderate latency at the smallest cost in terms of total number of forwardings. The total number of forwarding is high when T is small because most of the nodes are included in a very small number of communities, and the interest lists are broadcast in the community. When T is close to 1, most of the communities will contain only one node which makes the MOPS scheme downgrade to the Neighbors scheme. We also compare the choice of the default weight in this setting. With the default value 0, the events that are not matching a brokers' own and one-hop neighboring communities' interests won't be collected and stored. Thus, the MOPS scheme shows larger latency in this case, with a smaller number of forwards. The situation is opposite when the default value > 0 (10^{-5} in the simulation).

We investigate the performance of different schemes with varying event generation rates and the result is shown in Fig. 11. Both the efficiency and utility for all schemes decreases when the event generation rate increases. Some events may not reach the interested nodes when the event generation rate is high because the contact length between nodes is limited. However, we see that the curves of the MOPS scheme are less affected by the increasing in event generation rate, since the weight guided inter-community communication reduces the waste of precious bandwidth in DTNs.

In summation, MOPS outperforms the Neighbors scheme and simple strategies pure push and direct pull when we consider the utility and efficiency at the same time. Improvement is consistently shown in scenarios with different contact patterns. This indicates that MOPS can achieve satisfactory performance in a variety of DTNs. Considering that nodes only need local information of limited hops to form communities and determine the weight of each interest category, MOPS is certainly an efficient distributed pub/sub scheme for providing content-based service in the DTN.

VI. Conclusion

In this paper, we seek to utilize the community structure, which is based on long-term neighboring relationship between nodes in the DTN, to efficiently implement the pub/sub service. We define the similarity metrics based on nodes' encounter history to depict the neighboring relationship between nodes. The community is defined as a clique of nodes where any neighboring relationship is stronger than an adjustable threshold. Brokers are then deployed as the interface to match the interests and events among communities. The brokers utilize a unique weighted scheme to propagate interests and collect events. Extensive real- and synthetic-trace-driven simulation results are presented to support the effectiveness of

MOPS. In the future, we plan to study the weighted event propagation for pub/sub services in DTNs. Events will be attached with time-variant weight and the event exchange is based on both the aggregated interests and event weight.

REFERENCES

[1] P. Marshall. From self-forming mobile networks to self-forming mobile content services. *ACM Mobicom Keynote Speech*, 2008.

[2] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proc. of ACM SIGCOMM*, 2004.

[3] H. Jafarpour, S. Mehrotra, and N. Venkatasubramanian. A fast and robust content-based publish/subscribe architecture. In *Proc. of IEEE NCA*, 2008.

[4] E. Yoneki and J. Bacon. Distributed multicast grouping for publish/subscribe over mobile ad hoc networks. In *Proc. of IEEE WCNC*, 2005.

[5] Q. Yuan and J. Wu. Drip: A dynamic voronoi regions-based publish/subscribe protocol in mobile networks. 2008.

[6] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2006-09-15). http://crawdad.cs.dartmouth.edu/cambridge/haggle, September 2006.

[7] N. Eagle and A. Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

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

[9] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proc. of ACM SIGCOMM Workshop on DTNs*, 2005.

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

[11] A. Lindgren and A. Doria. Probabilistic routing protocol for intermittently connected networks. *draft-lindgren-dtnrg-prophet-03*, 2007.

[12] A. Balasubramanianm, B. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *Proc. of ACM SIGCOMM*, 2007.

[13] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of ACM MSWiM*, 2007.

[14] I. Leontiadis. Publish/subscribe notification middleware for vehicular networks. In *Proc. of Middleware doctoral symposium*, 2007.

[15] I. Carreras, F. Pellegrini, D. Miorandi, D. Tacconi, and I. Chlamtac. Why neighbourhood matters: interests-driven opportunistic data diffusion schemes. In *Proc. Of ACM CHANTS*, 2008.

[16] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.

[17] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proc. of ACM MobiHoc*, 2007.

[18] W. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy. Modeling time-variant user mobility in wireless mobile networks. In *Proc. of IEEE INFOCOM*, 2007.

[19] N. Djukic, M. Piorkowski, and M. Grossglauser. Island hopping: Efficient mobility-assisted forwarding in partitioned networks. In *Proc. of IEEE SECON*, 2006.

[20] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay tolerant networks. In *Proc. of ACM MobiHoc*, 2008.

[21] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proc. of MobiArch*, 2007.

[22] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[23] J. Bondy and U. Murty. *Graph theory with applications.* American Elsevier Publishing Company, 1976.

[24] P. Marsden. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4):407–422, October 2002.

[25] Y. Huang and H. Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. 2003.

APPENDIX

Local community construction. To construct the community, each node collects $(2k+1)$-hops of information, which can be collected via $2k$ rounds of encounter information exchange among neighbors. In the l-th round, a node sends its l-hop neighbor set and corresponding closeness value in the hello message to each of its encountered neighbors in this round. After collecting closeness from all its encountered neighbors in that round, the node now has $(l+1)$-hops of information.

Algorithm 1 Local community formation

for each node i in $N_k(init)$ **do**
  if $i \notin$ any clique formed by $init$ **then**
    $init$ starts a new clique with$\{init, i\}$;
    for each node j in $N_k(init)$ **do**
      if for $\forall l \in$ current clique, $j \in N_k(l)$ **then**
        Add j to current clique;
      end if
      Calculate $Ncut(A, \bar{A}) = \frac{\sum_{i\in A, j\in \bar{A}} c_{ij}}{\sum_{l\in A}\sum_{v\in G, v\neq u} c_{uv}}$;
    end for
  end if
  Choose the clique with minimum $Ncut(A, \bar{A})$ value;
  Set the label of this clique as the ID of initiator;
  Inform initiators' visible neighborhood $N_k(N_k(init))$;
end for

Each node i constructs its local neighborhood $N_k(i)$. Each node in $N_k(i)$ has a direct link or virtual link (including paths up-to-k-hops) with closeness $c \geq T$. Node i also knows $N_k(N_k(i))$, which includes all the nodes and direct or virtual links in its local neighbors' local neighborhood.

In MOPS, a node should select itself as the initiator if its degree d_i is the highest within its visible neighborhood $N_k(N_k(i))$. If two nodes with the same degree are visible to each other, node ID will be used to break the tie. Each initiator $init$ performs Algorithm 1 to form a *local community* with the nodes in $N_k(init)$.

Algorithm 1 aims to maximize the communities while keeping strong links inside the community. Thus it adopts the normalized cuts in the k-hop neighborhood division. Note that the local normalized cut value $Ncut(A, \bar{A})$ is computed based on the weights of the direct links, which are based on the initiator's $2k$-hop information. Here $\bar{A}$ denotes nodes in the $N_k(N_k(init))$ range but outside the community.

The initiator should propagate the formed community to its local neighbors. Nodes that haven't been included in the communities will exclude nodes in the formed communities and continue with local community formation process.

Betweeness calculation. The betweenness centrality of a gateway l in community A connecting the community B is calculated as follows:

$$b_l = \frac{\sum_{v\in B}(c_{lv} \cdot \sum_{i\in A}\sum_{j\in B}(c_{il} \cdot c_{vj}))}{\sum_{u\in A}\sum_{v\in B}(c_{uv} \cdot \sum_{i\in A}\sum_{j\in B}(c_{iu} \cdot c_{vj}))}, \quad (7)$$

A gateway l knows all other gateways in community A connecting to B, and also knows their centrality values from the $(2k+1)$ hops of local information collected in the community detection step. A gateway l calculates a measurement $b_{\bar{l}} = \sum_{i\in\{i|b_i>b_l\}} b_i$ to evaluate those gateways, which belong to the same community, have higher betweeness centrality than l, and connect the same pair of communities A and B.