Information-Centric Delay-Tolerant Mobile Ad-Hoc Networks

You Lu, Xiao Li, Yu-Ting Yu, Mario Gerla
Computer Science Department, UCLA
Los Angeles, USA
{youlu, pololee, yutingyu, gerla}@cs.ucla.edu

***Abstract*—Information-centric networks have recently been drawing increasing attention in academia as well as in industry. Information and content retrieval is a critical service for mobile ad-hoc networks. It relies on other resources and tools, such as internal storage, content searching and sharing, delay-tolerant delivery, etc. Previous studies have shown that social networking can assist delay-tolerant routing design in many respects. In this paper, we specifically address content retrieval in delay-tolerant mobile ad-hoc networks. We propose a social-tie based content retrieval scheme to support the delay-tolerant MANET. The social hierarchy is structured using balanced connectivity criteria and a *K*-mean clustering algorithm. The proposed scheme has been evaluated and validated on a real social network dataset.**

Keywords—social network forwarding; information-centric network; delay-tolerant mobile ad-hoc network

I. INTRODUCTION

Mobile ad-hoc networks (MANETs) are most effective in dynamic environments where network infrastructure is either not readily available or not adequate. MANETs can provide various services such as communication, storage, and computing for a range of applications. Sparse MANETs are a subclass of MANETs in which the node population is sparse, and contact between nodes in the network is infrequent. As a result, message delivery in sparse MANET must be delay-tolerant. One challenge in sparse MANET design is the ability to manage and serve the large amount of content distributed among different nodes.

Information-centric network (ICN) is an alternative to IP-based computer networks and is designed for content search and retrieval. In ICN, users only focus on the content they are interested, they need not know where the content is stored. Content retrieval follows the query-reply mode. The consumer broadcasts Interest packets through the network. When matching content is found either in the content provider or in an intermediate content cache server, the data is delivered tracing in reverse the "bread crumbs" left by the query.

One major design challenge of content retrieval in delay-tolerant MANETs is mobility. In sparse MANETs, network connectivity is highly dynamic and can vary rapidly. A common approach to deliver messages in a delay-tolerant network is social network routing. Disconnection gaps between nodes are overcome using a carry-and-forward principle. However, social network assisted delay-tolerant routing cannot be deployed directly in mobile ICN, since the content provider is not exposed during the content search phase, and a naïve-flooding query method will produce a high transmission cost. Moreover, content search and content delivery phase are sequential and, as we shall see, cannot use the same social network for routing.

In this paper, we propose a Social-Tie based Content Retrieval (STCR) scheme in delay-tolerant MANETs, which address the scalable content retrieval in large-scale sparse MANETs. In STCR, nodes record the essential information when they encounter each other. To compute a social graph of encounter relationships, we build a hierarchical architecture using *K*-mean clustering algorithm based on the social-tie between nodes. This hierarchy is used to improve forwarding strategy's efficiency. We also propose novel methods to compute the social-tie relationship considering both frequency and recency, and to compute centrality sequence considering both average social-tie and its balanced distribution. After the query is matched by one of the content digests, the search process will turn to social-tie routing to reach the content provider. We fully implement STCR in ns-3 simulator and evaluate its performance using real social network datasets in a sparse MANET experiment. We show that STCR scheme is effective in providing efficient content retrieval in delay-tolerant MANETs.

The rest of the paper is organized as follows. Section II covers related work, section III describes the design of the proposed scheme in detail, section IV presents the results of performance evaluation, and section V concludes the paper.

II. RELATED WORK

In this section, we introduce the general concept of ICN, and review delay-tolerant network and routing schemes.

A. Information-Centric Network

Information-centric network is an alternative approach to the architecture of IP-based computer networks. The basic principle is that user only needs to focus on content of interest, rather than having to reference a specific, physical location where the content is to be found. ICN differs from IP-based routing in three aspects. First, all contents are identified or named by a well-defined naming scheme. Second, caching is offered through the entire network to help the content distribution. Third, packet communications follow the query-reply mode. User (i.e., content consumer) spreads his interested content name as the "Interest" packet through the network. When one "Interest" packet hits the content in intermediate cache or the media server (i.e., content provider), the content packets will be forwarded back to the content consumer along the reversed Interest forwarding path.

A number of previous studies focused on ICN architectures and provided sketches of the required components. Content-centric network (CCN) [1] and named data network (NDN) [2] are two popular ICN proposals. Their components including Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store.

There are clearly many potential challenges that remain to be appropriately analyzed and integrated into ICN architectures for mobile scenarios. One prominent example is the need for delay-tolerant forwarding, a function that is increasingly important considering the easily disrupted mobile communications.

B. Delay-Tolerant Network

A delay-tolerant network (DTN) is a network architecture that supports significant delays or disruptions between data search and data receive phases [3]. Using the carry-and-forward model, DTN will temporarily store and carry the data during network disruptions until an appropriate "next hop" can be reached in network. These disruptions and delays could be caused by a number of reasons such as low density of nodes, network failures, and wireless propagation limitations. One extensively studied example of DTN is the Pocket Switched Network (PSN) [4]. PSNs exploit opportunistic human contacts for transmission and routing, typically by creating ad hoc links between neighbor mobile phones.

Different DTN routing protocols have been proposed in the past. The key observation is that encounters in a real environment do not occur randomly [5] and that nodes do not have an equal probability of encountering each other. Hsu et al claim that pedestrians never encounter more than 50 percent of the overall population [6]. As a consequence, nodes must assess the probability that they will encounter the destination node. It was found that node encounters are sufficient to build a connected relationship graph, which is a small-world graph. In this type of graph, a node does not send messages to neighbors

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

randomly, rather it sends messages to the neighbor perceived as the best carrier to destination based on gathered information [7].

Previous work studied various types of information that can be used to help packet delivery, including historical contacts, device mobility patterns, and social interaction between the participants. PeopleRank [8] is a typical social network based forwarding scheme. It is inspired by the PageRank algorithm [9] used in Google's search engine to measure the relative importance of a Web page within a set of pages. PeopleRank identifies the most popular nodes to forward the message to, given that a higher PeopleRank value means more "central" in the social graph. The explicit friendships are used to build the social relationships based on their personal communications. One potential problem in PeopleRank is that it is difficult to guarantee that the social graph from friendship and the one induced by the physical connection network are always consistent. The performance of PeopleRank highly depends on how the social graph is built.

SimBetTS [7] attempts to uncover a social network structure in DTN, using egocentric centrality and its social similarity to forward messages toward the node with highest centrality to increase the possibility of finding the potential carrier to the final destination. Although SimBetTS considers both the closeness and betweenness aspects, these two aspects reflect the social relationship and direct the messages toward higher centrality to improve the probability of meeting the destination. The social metrics proposed in SimBetTS can be improved further to implement more reasonable forwarding strategies.

BubbleRap [10] beats all above routing schemes as claimed by the author in terms of accuracy and efficiency. It combines the observed hierarchy of centrality and observed community structure with explicit labels, to decide on the best forwarding nodes. In BubbleRap, the centrality is calculated by the prior flooding experiment first. In order to make it practical, BubbleRap computes the collected data within either 6-hour S-window or cumulative C-window. It proved empirically that past contact information can be used in the future estimation. But all the DEGREE or RANK data collected from the past contact information only reflect the frequency. The S-window method can represent a kind of recency information but decrease the accuracy of the computing by lacking the data before the 6-hour window.

All the above studies attempt to build a hierarchical centrality structure and forward packets toward the higher centrality node. The higher centrality value only means that this node has a higher average probability to meet all other nodes in network other than the lower centrality nodes. If the destination of the message is known, this average probability of higher centrality may not provide a better successful rate of packet delivery for that specific destination since an average encounter probability may not be consistent with the encounter opportunity to one specific node. On the other hand, it is obvious that not all of the higher centrality nodes can provide equivalent benefit during each forwarding. It is necessary to identify which nodes improve utility and which do not even though their centrality is higher than the current node.

Our work differs from all studies above in several aspects. First, we build the social hierarchy by centrality using both frequency and freshness criteria and employing a novel social metric computation. Second, we use centrality to direct the content query process to avoid naïve flooding method. After the content provider is exposed, we use social-tie routing to forward messages to the relay node that has the best probability to reach the content provider than other nodes. We don't use the concept of centrality in this phase since higher centrality only reflects the higher average probability to meet all other node, not the target destination. Third, we use the K-mean clustering algorithm to identify which forwarding direction offers the best performance in terms of message retransmission costs.

III. PROTOCOL DESIGN

In this section, we first give the common assumptions which drive some of design decisions of the protocols. After that, we describe the design of our STCR scheme in detail.

A. Assumptions

In this paper, the following assumptions are made.

- We assume the connection associated with each encounter is bi-directional thus facilitating two way communications during the period of encounter.
- In the given topology, node-id is used as the unique identifier for a given node.
- We follow the naming scheme in NDN [2].
- Requests can be made randomly for any of contents from any of nodes.

B. Preliminaries

In this section, we provide the basic definitions for our protocol design. A sparse MANET is a network where the node density is not high enough to instantly connect all the nodes. Thus, packets must be forwarded in carry-and-forward mode if necessary. When there is no path from source to destination, a node will hold the packet until it encounters another node that has a higher possibility than itself to deliver the packet to the destination. This delivery mechanism causes a significant delay from sender to receiver, and is only suitable for applications with no real-time requirements.

To support the content retrieval in sparse MANETs, STCR performs the following main operations.

1) *Advertise Hello message*

Each node periodically advertises *Hello* messages for discovery of encounters in its transmission range. *Hello* message contains sender's node-id. *Hello* transmission interval is 100ms.

2) *Record encounter event*

A data structure called encounter-vector which includes encountered node-id and timestamp of this encounter event, as shown in (1), will be created after each encounter event:

<node-id, timestamp> (1)

Every node maintains an encounter-table which stores encounter-vectors created by the node at the encounter time.

3) *Compute social relationship*

The social relationship is the results of the aggregation of several indicators. Previous proposals have included metrics such as online social network graph [8], mutual friends [7], community label, friends connection duration [10], etc. All of them can reflect the social relationship in some aspects. In human networks, intuitively, if I have a message for a certain receiver, I will try to find a relay person who knows more people than I, since a more sociable person has a higher probability to see the receiver in the near future. This is the reason why all previous studies have built a social hierarchy based on the centrality relationship stemming from either historical contacts or human social graph. The higher centrality represents a higher average encounter probability to all other nodes. However, it is natural to assume that if two people have met frequently in the recent past, they must be in a close relationship. This recent encounter history can be used to predict the near future. BubbleRap [10] has shown that past contact information can be used to accurately predict future contacts. Frequency and recency become then the fundamental factors which must be monitored to build other metrics, e.g., social graph and mutual friends. Inspired by SimBetTS [7], we use frequency and freshness to describe social relationship.

Frequency metric is used to evaluate how frequently two nodes meet each other. We think two nodes have a strong relationship if they meet frequently. Freshness metric is used to evaluate the encounter's timestamp distribution reflecting how recently nodes have met each other. A strong social relationship stems from recent rather than remote encounters. Thus, we value recent encounter events higher than older ones. Combining the concepts of frequency and freshness, we define the social-tie concept that will be used to evaluate two nodes' social relationship.

Inspired from LRFU [11], each node computes a social-tie value to evaluate its relationship with other node and prioritize those relationships. As discussed earlier, the social-tie value is derived from the encounter history. The encounter event's contribution to this value is determined by a weighing function $F(x)$, where x is the time span

from the encounter event to the current time. Assume that the system time is represented by an integer and based on n encounter events of node i, the social-tie value of node i relationship with node j at time t_{base}, denote by $R_i(j)$, is defined as (2).

$$R_i(j) = \sum_{k=1}^{n} F(t_{base} - t_{j_k}) \quad (2)$$

where $F(x)$ is a weighing function and $\{ t_{j_1}, t_{j_2}, \cdots, t_{j_n} \}$ are the encounter time when node i met node j and $t_{j_1} < t_{j_2} < \cdots < t_{j_n} \leq t_{base}$.

For example, assume that node *i* met node *j* at time 1, 3, and 5 and the current time (t_{base}) is 10. Then node *i* 's social-tie to node *j* at t_{base}, denoted by $R_i(j)$, is computed as

$$R_i(j) = F(10-1) + F(10-3) + F(10-5) = F(9) + F(7) + F(5)$$

We take $F(x) = (\frac{1}{2})^{\lambda x}$ where λ is a control parameter and $0 \leq \lambda \leq 1$ as the weighing function, which have been proved in [11]. First, this control parameter allows a trade-off between freshness and frequency in contributing to the social-tie value. As λ approaches 0, frequency contributes more than freshness. When λ equals to 0, the social-tie value is simply derived from frequency. On the other hand, as λ approaches 1, freshness has much more effects on the social-tie value than frequency. When λ equals to 1, the social-tie value is simply determined by freshness. Following the example in [11], we set $\lambda = 1e^{-4}$. Second, suppose that node i has n encounter events with node j, the social-tie value at the time of nth encounter event can be computed from the time of the $(n-1)th$ encounter event and the social-tie value at that time. The computational and storage overhead can be reduced drastically due to this feature and each node is not required to maintain the record of all the past encounter events. Third, as discussed before, for each node, the relationships with other nodes should be prioritized or ordered according to the social-tie values. Due to the property of social-tie value definition and the feature of the above weighing function, the order of the social-tie values does not change until a new encounter event occurs, i.e., the order of the social-tie values changes only if there is a new encounter event. Hence, reordering of the social-tie values is needed only upon a new encounter event, though the social-tie values change over time.

4) *Exchange social-tie table*

Each node maintains a social-tie table that contains the social distances from the current node to all other encountered nodes, and each social-tie comes with a timestamp t_{base} when computed. During the encounter period, the social-tie table is exchanged and merged into the other node's social-tie table. This process is similar to a routing update in link-state routing protocol. When a node receives a social-tie table from other nodes, it will refresh the local social-tie table according to the timestamps. Eventually, a social-tie table in a node will contain all the nodes' social-tie in network, but social-tie table convergence progress is very slow due to the long latency feature of DTN.

5) *Compute centrality*

Based on the social-tie table, a node can compute each node's centrality. Centrality measures the average social distance from the given node to all other encountered nodes. The centrality can be regarded as a measure of how long it will take information to spread from a given node to all other nodes in the delay-tolerant network. It is obvious that the average social-tie from the given node to all other nodes can be computed as (3), where *N* is the number of nodes observed from social-tie table, and *R* is the social-tie from the given node to each of other nodes.

$$\frac{\sum_{k=1}^{N} R_i(k)}{N} \quad (3)$$

However, in some case a node may have an unevenly distributed social-ties to other encountered nodes. This may also cause a relatively high average social-tie result. Obviously, the average social distance we preferred in network depends not only on the average social-tie values, but also on its social-tie's distribution. In delay-tolerant MANETs, the connectivity of the networks is very dynamic. The high centrality node should not only have a high average social-tie value, but also have a high chance to encounter more other nodes. Accordingly, the social-tie distribution should be balanced. We propose a centrality estimation method that considers both the average social-tie values and their distribution to achieve higher degree of centrality. We adopt Jain's Fairness Index mechanism [12] to evaluate the balance distribution of social-tie values. As in equation (4), Jain's Fairness Index is used to determine whether users or applications are receiving a fair share of network resources.

$$Jain's\ Fairness\ Index: balance = \frac{(\sum x_i)^2}{n \times \sum {x_i}^2} \quad (4)$$

Jain's equation rates the balance of a set of values. The result ranges from 1/n (worst case) to 1 (best case). Jain's metric identifies underutilized channels and is not unduly sensitive to a typical network flow pattern. In our approach, Jain's fairness index is used to evaluate the balance of social-tie connection. The centrality metric is defined in (5), where *N* is the encountered node count in the encounter table.

$$C_i = \alpha \frac{\sum_{k=1}^{N} R_i(k)}{N} + (1-\alpha) \frac{\left(\sum_{k=1}^{N} R_i(k)\right)^2}{N \times \sum_{k=1}^{N} (R_i(k))^2} \quad (5)$$

Here α (set in our experiment as 0.5) is a parameter decided by the user according to the specific scenario and network conditions. For example, if there are few nodes in a large area with high mobility, we prefer a smaller α since here the balanced connection opportunity between nodes is more important. Otherwise, if more nodes exist in a relatively small ground, we may consider a bigger α value. A higher centrality value means that the node has been meeting other nodes more often and more recent in network.

C. Content Name Digest

Content management in content retrieval application is a challenge. Using a plain content name list is costly and not scalable when the network and number of content become large. In our proposed scheme, the content names are advertised in the form of content name digest. Bloom Filter is a technique to map a content name list to a bit vector, in which the content name verification operation can be carried out within $O(1)$ operations instead of $O(m)$ operations (m is the content count) required by the plain content name list. When a Bloom Filter is used to represent the content name list of a node, the size of the digest advertised by the node is much smaller than plain content names. Thus, our proposed scheme becomes more scalable. Fig. 1 shows the construction of the Bloom Filter according to a plain content name list. A bit vector of *m* bits is used to represent a set of n contents $\{C_1, C_2, \ldots, C_n\}$. Originally all the bits in the Bloom Filter are set to "0". After hashing each item using a hash function of $log_2(m)$ bits, the bits are added to the Bloom Filter and set the corresponding bit to "1". To check the membership of the element x, it is sufficient to verify whether the bit corresponding to *h(x)* is set to "1". The verification will cause "false positive", i.e., an element not belonging to the set may be checked as a member. But Bloom Filter is free from false negative, i.e., any element verified as a non-member shall not belong to the set. Many hash functions such as MD5 and SHA-1 are evenly distributed in the "bit vector" domain. According to the results from [13], we set *m* as 128 bits to achieve moderate false positive.

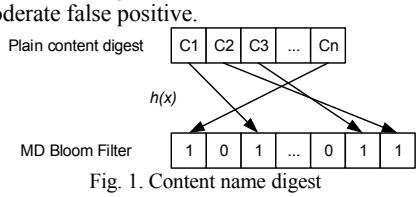

Fig. 1. Content name digest

D. Digest Convergence

A node can compute each observed node's centrality from social-tie table and form a sequence of social relationship among the observed nodes in network. The higher centrality node has a higher probability to meet other nodes than the lower level node. In the content query phase, in order to avoid pure flooding, we design this digest convergence process. The basic idea is that each content provider actively announces its content name digest to higher centrality nodes. When a node encounters another higher centrality node, it will send its content name

digest with the timestamp to that node. Each node maintains a local data structure called digest table, as shown in TABLE I, to store the received digests from lower centrality nodes.

TABLE I. DIGEST TABLE

Provider ID	Digest	Timestamp
Node-2	Digest-2	Time-2
Node-3	Digest-3	Time-3
…	…	…

And then, the digest table will be sent to another higher centrality node encountered later, as shown in Pseudo code 1. If received multiple copies of content name digest of the same content provider, the digest table is updated according to the freshest timestamp. In this way, each node collects content name digests from lower centrality nodes, and reports the collected digests to higher centrality nodes. Thus, the content name digests from each content provider are converged towards the higher centrality nodes in network. The higher centrality node has the larger knowledge of the content name digests, and knows which content provider contains which content.

Pseudo code 1: Digest Convergence Process

```
1: when receiving a content digest from other node
2:     insert this digest into my local content table
3: when encountering a node
4:     if (his social hierarchy is higher than me) then
5:         send my local content table including my content digest to him
6:     else
7:         do nothing
8:     end if
```

E. Content Request

When a content requester wants to request a certain content, an Interest packet which contains requester's node-id and content name will be generated and forwarded to a higher centrality node to avoid naïve flooding, because higher centrality node has more knowledge on content name and content providers. Each node can compute the centrality of the newly encountered node from local social-tie table. If we compute the centrality of each node in social-tie table and sort then in order, we will find that the interval of centrality is not even, as shown in Fig. 2.

Fig. 2. Centrality sequence

If the relay node has a similar centrality with the current node, they may have a similar knowledge on the content name digests, thus we may not get much benefit from this forwarding. Intuitively we prefer a relay node whose centrality has enough difference than that of current node, to further reduce transmission cost. Inspired by clustering algorithms, periodically, we divide nodes into clusters according to their centrality distribution, and forward the Interest packet to a newly encountered node which belongs to a higher centrality cluster, as shown in Fig. 3. The Interest packet is only forwarded from cluster A to cluster B. There is no Interest forwarding within a cluster.

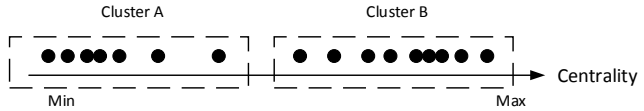

Fig. 3. Centrality Clusters

1) Centrality clustering

We use *K*-mean clustering algorithm to define clusters. *K*-means clustering is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. Given a set of observations ($x_1, x_2, \ldots, x_n$), *k*-means clustering aims to partition the *n* observations into *k* sets ($k \le n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS), as shown in (6), where μ_i is the mean of points in S_i.

$$\arg min_s \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - u_i \right\|^2 \qquad (6)$$

Here *K* (set in our experiments as 10) is a parameter that is decided by user according to the specific scenario and network scalability. The larger *K* value will benefit the packet delivery ratio but cause higher transmission cost. The nodes in the same cluster form a social level in centrality hierarchy. The packet will be forwarded to upper social level in our forwarding strategy.

2) Interest packet forwarding

As described above, we use K-mean clustering algorithm to build a social hierarchy, and nodes in social-tie table are assigned into different levels. The requester carries the Interest packet and forward it to the first encountered node that has a higher social level than itself. After that, the requester keeps a copy of Interest packet and forwards to the next encountered node that has an even higher social level than the relay node it forwarded to last time. After a node receives the Interest packet from other nodes it encountered, it will first check its local digest table to see if there is any matched name. If no matched name is found, it will continue forwarding the Interest packet. Each relay node performs the same strategy: forwarding the Interest packet to the next relay node that has a higher social level than the last relay node. This is because, after the last forwarding, if the node meets a better relay node which has a higher social level than last one, forwarding the Interest packet to this new relay node will get more benefit. Following this strategy, the Interest packet is forwarded upward level by level or jumps to a higher level towards the most popular node in the centrality hierarchy, as shown in Pseudo code 2.

Since the content name digest keeps being updated and converges toward the higher social level nodes, the query of Interest passing toward the higher social level in the network will be solved eventually when the Interest name matches one content name in the digest table for a certain node at some level of the hierarchy. At this point, the Interest packet will turn into social-tie routing toward the destination since the content provider id has now been disclosed.

There is a potential problem caused by DTN. Similar with the convergence issue in a link-state routing protocol, due to the carry-and-forward scheme in DTN, the social-tie table convergence suffers from a significant delay, which causes the problem that the information used to compute the centrality sequence is not consistent between nodes. In order to make the design practical, we build the social relationship in a distributed method and the computing result comes from node's local database (i.e., social-tie table). When the encounter happens, the local social-tie table gets updated to refresh the social relationship result. This can be treated as a learning phase while the social relationship becomes more accurate during each update. Since the previous contact information can be used to predict the future encounter, and the social-tie table grows to be more accurate, the impact of inconsistent social-tie table diminishes and can be tolerated as time progresses. However, there may still be routing loop due to the inconsistent centrality sequence.

Pseudo code 2: Forwarding Strategy

```
1: when having an Interest packet
2: check my local content table
3: if there is a match, then
4:     turn into social-tie routing
5: else
6:     keep this Interest packet
7:     loop: when encountering a node
8:     if (his social level is higher than me) then
9:         if (last_relay_node == null) then
10:            forward the Interest packet to him
11:            update last_relay_node
12:        else
13:            if (his social level is higher than last_relay_node) then
14:                forward the Interest packet to him
15:                update last_relay_node
16:            else
17:                do nothing
18:            end if
19:        end if
20:    else
21:        do nothing
22:    loop back
22: end if
```

A Time-To-Live (TTL) setting is configured in Interest packet and counts down during content query phase. And a waiting timer is setup by content requester after sending out the Interest packet. Routing loop will cause the TTL gets zero and requester's waiting timer runs out. In this case, we provide a fallback forwarding strategy that the relay node always delivers the Interest packet to a higher social level node than itself, not the higher social level than the last relay node in the previous method, as shown in Pseudo code 3.

Pseudo code 3: Fallback Forwarding Strategy

```
1: when having an Interest packet
2: check my local content table
3: if there is a match, then
4:     turn into social-tie routing
5: else
6:     keep this Interest packet
7:     loop: when encountering a node
8:     if (his social level is higher than me) then
14:        forward the Interest packet to him
19:    end if
22:    loop back
22: end if
```

If the fallback forwarding strategy is adopted and the waiting timer runs out again, the content requester will start the epidemic routing [14] to flood the Interest packet throughout the network. The header of Interest packet has a forwarding flag to indicate one of three forwarding strategies set by content requester, and each relay node follows the specific forwarding strategy during the forwarding process.

F. Social-Tie Routing

In this step, the content provider's node-id has been disclosed and attached in the Interest packet. So the task is to forward the Interest packet to the destination. Similar with the centrality sequence, using the local social-tie table and K-mean clustering, we can generate a content provider's social-tie sequence and build a social-tie hierarchy. The relay node then forwards the Interest packet to the newly encountered node who has a higher social-tie level to the destination node compared to its own, and follows the three forwarding strategies indicated by the flag in the Interest packet header. In summary, in the content query phase described in section E, an Interest packet is forwarded toward higher centrality level since a higher centrality node has more knowledge of content name digest in network. After Interest matches a content name digest, we turn to the Interest packet delivery phase which forwarding Interest packet to the content provider. In this phase, Interest packet is forwarded toward higher social-tie level of content provider since a higher level node is closer to the destination.

After the Interest packet eventually reaches the content provider, the content provider will send the content back to the requester using the social-tie routing again and copy the forwarding flag from the Interest packet header. Each relay node will compare newly encounter node's social-tie of the content requester and forward the content toward higher social-tie level of the content requester. The content provider only responses once to the same Interest packet and requester, and ignores the following received duplicate Interest packets. In the content data retrieval period, the destination node-id is the content requester. The content data packets will be forwarded by the same forwarding strategy as the Interest packet has.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed STCR scheme in a packet-level simulation with the real mobility trace dataset.

A. Simulation Setup

We implemented the proposed STCR scheme in NS-3.17 network simulator. DTN nodes advertise their *Hello* message to each other every 100 ms. In order to test the bottom line of the performance, we assume that each node has a unique content which is different with all other nodes. Periodically (every 30 seconds), a random node is selected to generate an Interest packet for any content. We also assume that the content data can be retrieved as 1MB size so that the measurement will not be affected by the content size variance.

We use the IEEE 802.11g wireless channel model and the PHY/MAC parameters as listed in TABLE II. To gain meaningful results, we use INFOCOM'06 contact traces taken from 98 nodes during the INFOCOM'06 conference in 95 hours [15].

B. Evaluation Metrics

To abstract away from any particular routing algorithm, we use the Epidemic routing [14] in an attempt to understand the upper bounds of connectivity (epidemic has the highest delivery probability). In Epidemic, when two nodes meet each other, they exchange messages that they haven't seen. This scheme makes the Epidemic routing creates unlimited number of messages by copying the messages to all nodes that do not yet have to copy. We also compare our design to PeopleRank and BubbleRap which is advanced to SimBetTS as claimed in [10]. We use the following metrics in the experiments.

1) Hit rate: the percentage of Interests that are successfully delivered to the content providers and the content data are successfully delivered to the requesters. This metric reflects the capability of a method to discover the requested content.

2) Average delay: the average delay time of the successfully delivered content from the Interest has been sent out. This metric reflects the efficiency of a method to discover the requested content and retrieve it back.

3) Total cost: the total number of messages replicas in the network. To normalize this, we divide it by the total number of unique messages created.

TABLE II. SIMULATION PARAMETERS

Parameter	Value
RxNoiseFigure	7
TxPowerLevels	1
TxPowerStart/TxPowerEnd	12.5 dBm
m_channelStartingFrequency	2407 MHz
TxGain/RxGain	1.0
EnergyDetectionThreshold	-74.5 dBm
CcaMode1Threshold	-77.5 dBm
RTSThreshold	0 B
CWMin	15
CWMax	1023
ShortEntryLimit	7
LongEntryLimit	7
SlotTime	20 μs
SIFS	10 μs

C. Experiment Results

1) *Hit rate*

Fig. 4 shows the hit rates over time in different methods. The parameter K is set to 10. We find that Epidemic can retrieve most requested contents while our proposed STCR scheme finally reaches about 68% which is better than BubbleRap and PeopleRank. Epidemic has the highest hit rate because a node copies its Interest to all other nodes, which will eventually hit the content provider in highest probability. The performance of STCR is worse than that of Epidemic method, but still better than that of BubbleRap and PeopleRank, which shows the social-tie routing scheme is more efficient than BubbleRap's centrality + community scheme and even better than PeopleRank's centrality-only scheme.

2) *Average delay*

Fig. 5 illustrates the average delay over time of the three methods. From the figure we can see that the delay of Epidemic is much lower than other three methods. We only measure the delay of successful content retrieval. In Epidemic, Interest is rapidly flooded in network. As a result, the Interest can reach its destination after a short delay. Both BubbleRap and PeopleRank show larger delays than STCR because both Interest and content packets are only forwarded to the higher social centrality nodes which causes a longer waiting time both in content query phase and content retrieval phase. And PeopleRank gets the highest delay since it only uses centrality without any community information as BubbleRap.

3) *Total cost*

Fig. 6 shows the total cost of different methods. The cost of Epidemic exceeds all other protocols. In Epidemic, each node delivers its Interest or content packet with every encountered node, which contributes to the highest cost. In STCR, each node exchanges its social-tie table with other newly encountered nodes, but only deliver the Interest and content packet to selected relay nodes using the clustering method. So, in terms of transmission cost, STCR is better than BubbleRap and PeopleRank.

4) *Impact of K-mean*

The parameter *K* is set to 10 in above experiments. A higher *K* will induce more clusters in network, thus generate more packet forwarding which benefits Hit Rate and Average Delay but hurts Total Cost. So the value of *K* should be selected carefully according to the specific scenario. We exam *K* with different value to show its impact to these evaluation metrics, as shown in Fig. 7-9.

V. CONCLUSION

Information-centric network has been drawn more and more attentions both in the academia and industry. Designing an efficient content retrieval scheme in large-scale delay-tolerant mobile ad hoc networks is one of its biggest challenges. Current ICN approaches generate excess transmission cost that causes scalability issues when network and content number grows. In this paper, we propose STCR, a social-tie based content retrieval scheme that is highly scalable in delay-tolerant mobile information-centric networks. The STCR generates the social-tie based routing structure in order to support an efficient Interest and content forwarding. We use K-mean clustering algorithm to build the social level forwarding scheme in order to reduce the transmission cost. We also propose some novel methods to compute social metrics considering both the frequency and freshness of encounters, and balanced connectivity with all other nodes to improve the delivery rate. We implemented our scheme in NS-3 simulator and show that STCR scheme is effective in providing efficient content retrieval in delay-tolerant MANETs.

REFERENCES

[1] V. Jacobson, M. Mosko, D. Smetters, and J. Garcia-Luna-Aceves, "Content-centric networking," Whitepaper, Palo Alto Research Center, pp. 2-4, 2007.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, et al., "Named data networking (ndn) project," Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, 2010.

[3] K. Fall, "A delay-tolerant network architecture for challenged internets," presented at the Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, 2003.

[4] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding," Technical Report UCAM-CL-TR-617, University of Cambridge, Computer Laboratory, 2005.

[5] N. Sastry, D. Manjunath, K. Sollins, and J. Crowcroft, "Data Delivery Properties of Human Contact Networks," Mobile Computing, IEEE Transactions on, vol. 10, pp. 868-880, 2011.

[6] W.-j. Hsu and A. Helmy, "Impact: Investigation of mobile-user patterns across university campuses using wlan trace analysis," arXiv preprint cs/0508009, 2005.

[7] E. M. Daly and M. Haahr, "Social Network Analysis for Information Flow in Disconnected Delay-Tolerant MANETs," Mobile Computing, IEEE Transactions on, vol. 8, pp. 606-621, 2009.

[8] A. Mtibaa, M. May, C. Diot, and M. Ammar, "PeopleRank: Social Opportunistic Forwarding," in INFOCOM 2010 Proceedings pp. 1-5, 2010.

[9] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," Computer Networks and ISDN Systems, vol. 30, pp. 107-117, 4, 1998.

[10] H. Pan, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks," Mobile Computing, IEEE Transactions on, vol. 10, pp. 1576-1589, 2011.

[11] C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," IEEE transactions on Computers, vol. 50, pp. 1352-1361, 2001.

[12] R. Jain, D.-M. Chiu, and W. R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer system: Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[13] M. V. Ramakrishna, "Practical performance of Bloom filters and parallel free-text searching," Commun. ACM, vol. 32, pp. 1237-1239, 1989.

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

[15] http://crawdad.cs.dartmouth.edu/meta.php?name=cambridge/haggle/imote#N10036.

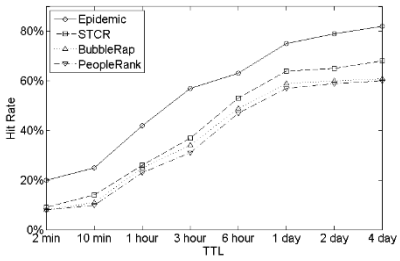

Fig. 4. Hit Rate

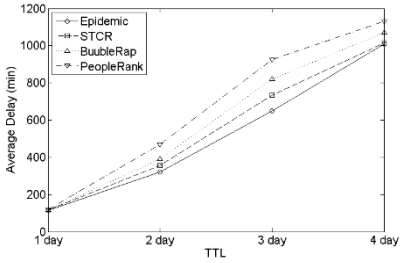

Fig. 5. Average delay

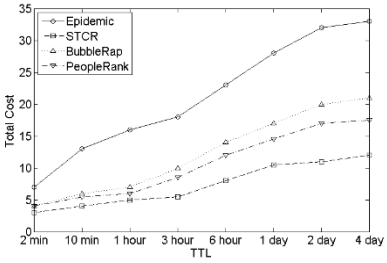

Fig. 6. Total cost

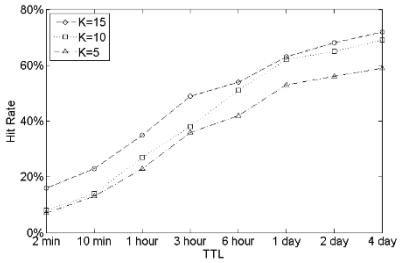

Fig. 7. Hit rate with different K

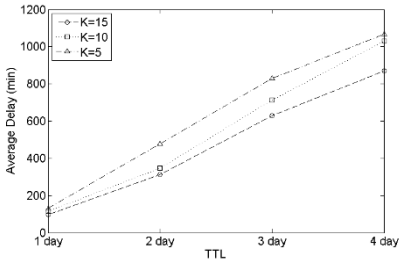

Fig. 8. Average delay with different K

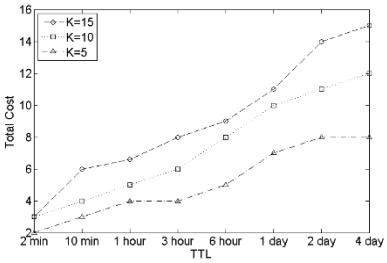

Fig. 9. Total cost with different K