# Expertise-Based Data Access in Content-Centric Mobile Opportunistic Networks

Jing Zhao*, Xiaomei Zhang*, Guohong Cao*, Mudhakar Srivatsa† and Xifeng Yan‡
*Pennsylvania State University, University Park, PA
†IBM T. J. Watson Research Center, Hawthorne, NY
‡University of California, Santa Barbara, CA
* {juz139,xqz5057,gcao}@cse.psu.edu, † msrivats@us.ibm.com, ‡xyan@cs.ucsb.edu

***Abstract*—In mobile opportunistic networks, most existing research focuses on how to choose appropriate relays to carry and forward data. Although relay selection is an important issue, other issues such as finding content from people with the right expertise are also very important since the ultimate goal of using mobile opportunistic network is to provide the right content to mobile users (nodes). In this paper, we study expertise-based data access in content-centric mobile opportunistic networks, where the objective is to minimize the average query delay given a sequence of queries considering node expertise, node queuing delay and communication delay. To solve this problem, we propose various query forwarding approaches under deterministic and probabilistic expertise models. Specifically, we propose centralized approaches to assign queries based on a modified Dijkstra's shortest path algorithm and distributed approaches in which query forwarding is based on a utility metric. Extensive simulations on both synthetic and realistic traces demonstrate that our solutions outperform existing approaches.**

## I. Introduction

Mobile opportunistic networks, also known as Disruption Tolerant Networks (DTNs) [1], consist of hand-held mobile devices such as smartphones, tablets, and laptops. The major advantage of mobile opportunistic network is that it does not rely on any infrastructure, and can be applied to battlefield, disaster recovery, environmental monitoring, habitat monitoring, transportation, and many cyber physical systems. In mobile opportunistic networks, due to node mobility and low node density, the network topology is highly dynamic and end-to-end connection is hard to maintain. To deal with these problems, researchers adopt the idea of carry and forward, where a node carries the data packet when no route exists, and later forwards the packet to a new node (relay) that moves into its communication range, referred to as *contact*. Then, the key problem in mobile opportunistic network becomes how to determine the appropriate relay selection strategy, and many researchers design different metrics for choosing the relays [2], [3], [4], [5].

Although relay selection is an important issue in mobile opportunistic networks, other issues such as finding content from people with the right expertise are also very important since the ultimate goal of using mobile opportunistic network is to provide the right content to mobile users (nodes). Consider a group of users in a disaster recovery area connected by mobile opportunistic networks. There are various kinds of information access traces left by these users, such as local documents, Web search, messages, emails, reports, and annotations, which are useful for decision making such as what is the evacuation plan? how to survive with limited source? how to help a wounded person? Different users and information sources possess different kinds of knowledge, and different kinds of expertise. It would often be the case that a user (say $v$) wants to learn something which falls into the expertise of another user. User $v$ may generate a query, and then route it through the network until it reaches some user who can respond it. Here routing is based on the content of the messages (queries), rather than explicit destination addresses assigned by the source. The key problem is hence how to design an effective routing approach to minimize the query delay in such content-centric mobile opportunistic network. Although there have been many prior works on data dissemination in mobile opportunistic networks [6], [7], [8], not much work has been done on expertise-based data access in content-centric mobile opportunistic network, which is the major focus of this paper.

The query delay is affected by several factors such as user (node) expertise, node queuing delay, and communication delay. When a node receives a query matching his expertise, the query is likely to be responded quickly, which means the processing delay is short. Otherwise, the user may have to forward the query to some other node, which increases the query delay. On the other hand, if a node has more than one query to be processed, some queries will be buffered, which incurs some queuing delay. The communication delay is determined by the node inter-contact time. In such content-centric mobile opportunistic networks, some simple query solutions may not work well. For example, flooding the whole network with the query may not necessarily reduce the query delay since it will increase the message overhead and increase the queuing delay. Thus, it is better to forward the query to the node with the right expertise and less pending queries, and thus it is a challenge to minimize the query delay by considering all these factors.

In this paper, we propose techniques to address the aforementioned challenges. We consider two node expertise models: the *deterministic expertise model* in which a query can be responded by all nodes but the processing time may differ from

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

one node to another, and the *probabilistic expertise model* in which a node can only respond a query with some probability. Under the deterministic expertise model, we first propose a centralized approach based on a modified Dijkstra's shortest path algorithm [9] by assuming that the query source has the up-to-date information (expertise profile, queuing delay, communication delay) of all nodes in the network. Note that since the network graph has both edge weight (communication delay) and node weight (node queuing delay and processing delay), we need to first transform the original graph to a new weighted graph before running Dijkstra's algorithm. Then, we propose a distributed approach since such up-to-date information is not always available. Under the probabilistic expertise model, we first propose a centralized approach in which the query source uses some pre-selected paths to route the query to a number of destinations which are likely to respond quickly. Then, we propose a distributed approach in which each node decides whether to process the query or to multicast the query to some other nodes, based on the collected information. Extensive synthetic and realistic trace driven simulation results validate the effectiveness of our approaches.

The rest of this paper is organized as follows. Section II reviews related work. Section III describes the system model and the problem formulation. We present various routing approaches under deterministic and probabilistic expertise models in Section IV and Section V respectively. Performance evaluations are presented in Section VI, and Section VII concludes the paper.

## II. Related Work

There have been a number of works to study data dissemination in mobile opportunistic networks [6], [7], [8]. They focus on forwarding data to the nodes which are interested in the data, whereas our work aims to forward queries to the nodes whose expertise is close to that required by the queries. For example, Gao et al. [6] proposed a user-centric data dissemination scheme which aims to use the minimum number of relays to forward data only to the nodes that are interested in the data. Lin et al. [7] designed another content dissemination protocol to maximally satisfy user preferences for various content objects, and later Guo et al. [8] also considered privacy issues in content dissemination. However, since these works do not consider the query processing time, and thus cannot be applied to address our expertise-based data access problem.

Recent years have witnessed the emergence of content-centric networking (also known as named data networking [10], [11], content-centric networking [12], [13] or data-oriented networking [14]), where the focus is on the content users wish to obtain instead of the servers that provide the content. As a result, the new communication architecture is built on named data and new approaches have been proposed to routing named content. However, query processing in our work may involve human in the loop and is much longer, which is not considered in existing content-centric networking.

There have been some existing works to study expertise-based data access in collaborative networks in which a group of people work together to achieve specific goals (such as resolving trouble tickets in IT service). Shao et al. [15] proposed a Markov model to guide ticket (query) forwarding by mining the ticket resolution sequences. However it does not consider the ticket content, and might not be able to find the best ticket routing sequences. To address these problems, Miao et al. [16] developed a more comprehensive model using both the content and the routing sequence of the ticket, which leads to better ticket forwarding decisions. Later in [17], they also studied the properties of collaborative networks (such as node degree distribution), and proposed a simpler model to capture how the tickets are routed by human beings. However, none of them consider the node queuing delay, and so the nodes with high expertise may be overloaded. They also do not consider the communication delay, which is a major factor in mobile opportunistic networks.

## III. Preliminaries

In this section, we first describe our system model, and then formulate the expertise-based data access problem.

### A. System Model

*1) Network Model:* We consider a mobile opportunistic network of $n$ users to process $m$ queries ($m >> n$). The network is modeled as a graph $G(V, E)$, where each node $v \in V$ corresponds to a user, and an edge $(u, v) \in E$ represents the stochastic contact process between a node pair $u$, $v$. Each edge is associated with the communication link delay, which is determined by the node inter-contact time. In disaster recovery, users (nodes) usually move in repetitive patterns (e.g., deliver supplies to evaluation camps, patrol given routes, report to bases). This indicates the node inter-contact time is relatively stable over time. Thus, we use the average inter-contact time between $u$ and $v$ (denoted by $c_{u,v}$) to represent the communication link delay of $(u, v)$.

*2) Expertise Model:* We consider both deterministic and probabilistic expertise models. Let $d$ be the number of expertise domains. Each query $q$ is described by a $d \times 1$ vector:

$$\mathbf{Q}_q = [w_{q1}, w_{q2}, \ldots, w_{qd}]^T \quad (1)$$

Here $[\cdot]^T$ indicates matrix transpose. $w_{qi}$ denotes the amount of expertise required by query $q$ in domain $i$ ($0 \leq w_{qi} \leq 1$), which can be specified by the node which generates the query. We have $\sum_{i=1}^{d} w_{qi} = 1$ for each query $q$.

**Deterministic Expertise Model:** For queries belonging to a specific expertise domain, they can be responded by all nodes but the processing time may differ from one node to another.

The expertise profile of node $v$ is a $d \times 1$ vector as follows:

$$\mathbf{P}_v = [t_{v1}, t_{v2}, \ldots, t_{vd}]^T \quad (2)$$

Here $t_{vi}$ denotes the processing time for node $v$ to respond a query belonging to domain $i$.

Let $T_v^q$ be the processing time for node $v$ to process query $q$, which may require expertise from more than one domain. $T_v^q$ is defined as follows:

$$T_v^q = \mathbf{P}_v^T \mathbf{Q}_q = \sum_{i=1}^{d} t_{vi} w_{qi} \tag{3}$$

**Probabilistic Expertise Model:** For queries belonging to a specific expertise domain, a node may not be able to respond all of them, and is even uncertain about whether an individual query can be responded or not before processing it. The expertise profile of node $v$ is a $d \times 2$ vector as follows:

$$\tilde{\mathbf{P}}_v = \begin{bmatrix} \mathbf{P}_v \\ \mathbf{A}_v \end{bmatrix}^T = \begin{bmatrix} t_{v1} & t_{v2} & \dots & t_{vd} \\ a_{v1} & a_{v2} & \dots & a_{vd} \end{bmatrix}^T \tag{4}$$

Similar to the deterministic expertise model, $t_{vi}$ denotes the processing time for node $v$ to respond a query belonging to domain $i$. The processing result is either success (node $v$ can respond the query) or fail (node $v$ cannot respond the query). We use $a_{vi}$ to denote the probability that node $v$ can respond a query belonging to domain $i$ ($0 \leq a_{vi} \leq 1$).

For a query $q$ which may require expertise from more than one domain, let $T_v^q$ be the processing time of node $v$, and $p_v^q$ be the probability of node $v$ to respond the query. $T_v^q$ is defined in the same way as Equation (3), and $p_v^q$ is defined as:

$$p_v^q = \mathbf{A}_v^T \mathbf{Q}_q = \sum_{i=1}^{d} a_{vi} w_{qi} \tag{5}$$

The probabilistic expertise model is more general than the deterministic expertise model, and the deterministic expertise model can be viewed as a special case of the probabilistic expertise model by assigning $a_{vi}$ to 1 for $\forall v, i$.

*3) Queuing Model:* If a node (user) is busy processing some query, it cannot process other queries at the same. In practice, each node has a queue to buffer pending queries which cannot be immediately processed. Queries in the queue are processed on a first-come-first-served basis. Once a node begins to process a query, it works without interruption until the query is processed. The queuing delay ($T_v$) can be calculated as the total processing time for all queries buffered in the queue.

### *B. Problem Formulation*

We formulate the expertise-based data access problem as follows.

*Definition 1:* ***Expertise-based data access problem***
Given $m$ queries, which need to be responded by $n$ nodes in a mobile opportunistic network $G(V, E)$. Node $v$ has probability $p_v^q$ to respond query $q$ and the processing time is $T_v^q$. Suppose query $q$ is generated by some source node at time $t_q$. Let $t'_q$ be the time that the source node receives the response for query $q$. The objective is to minimize the average query delay of all queries, i.e., $\min \frac{1}{m} \sum_q (t'_q - t_q)$.

In practice, it is difficult to predict the query generation time and the expertise requirement of future queries. Thus, it is a challenge to minimize the query delay. In the following two sections, we address this challenge by proposing various query forwarding approaches.

## IV. Query Forwarding Approaches under Deterministic Expertise Model

In this section, we first propose a Centralized Approach under the Deterministic expertise model *(CAD)* and then propose a Distributed Approach under the Deterministic expertise model *(DAD)*.

### *A. Centralized Approach under Deterministic Expertise Model (CAD)*

A query should be forwarded to the node that will provide response with the minimum delay. Let $D_v^q$ denote the query delay if node $v$ responds query $q$, which includes the communication delay, queuing delay, and query processing delay. The communication delay also includes the delay for routing the response back to the query source. Then, query $q$ should be assigned to node $v$ with the minimum $D_v^q$. In CAD, the source node of query $q$ uses a modified Dijkstra's algorithm to find the node $v$ with the minimum $D_v^q$, and then uses source routing to send the query to node $v$ along the path selected by the algorithm.

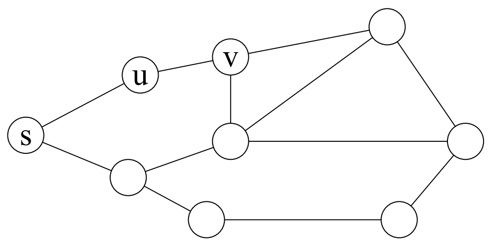


Fig. 1. An example network

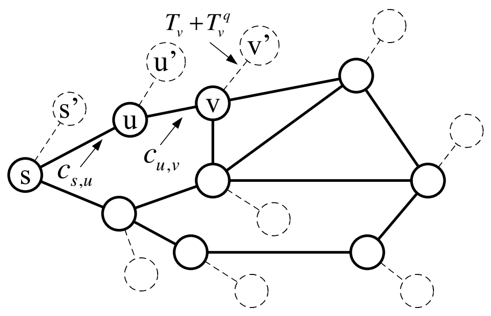


Fig. 2. A constructed weighted graph

Figure 1 shows an example network, in which each node corresponds to a user and an edge represents the stochastic contact process between a node pair. Suppose query $q$ is generated by node $s$. For each node $v$, $D_v^q$ may be modeled as the shortest path from node $s$ to node $v$ and then calculated using Dijkstra's algorithm. However, it is impossible to directly apply Dijkstra's algorithm here since the graph has both edge weight (communication delay) and node weight (node queuing delay and processing delay). Thus, we first transform the original graph to a new weighted graph (Figure 2), and then run Dijkstra's algorithm.

Specifically, for each node (called *real node*) in the original graph, a *virtual node* is added, which connects to the real node with an edge whose weight is the sum of the node queuing delay and the processing delay. For example, let node $v'$ be the virtual node of node $v$. Edge $(v, v')$ is associated with $T_v + T_v^q$ where $T_v$ is the queuing delay of node $v$, and $T_v^q$ is the processing time for node $v$ to process query $q$. Here, the query source is assumed to have all network information, and hence can obtain the accurate queuing delay of node $v$ when node $v$ receives the query. Other edges connecting two real nodes are associated with twice of the communication link delay, which also considers the delay for routing the response back to the query source. For example, if the average inter-contact time between $u$ and $v$ is $c_{u,v}$, the communication link delay of edge $(u, v)$ will be represented by $2c_{u,v}$. Then, $D_v^q$ is equal to the length of the shortest $s$-$v'$ path. The formal description of this modified Dijkstra's algorithm is shown in Algorithm 1.

*1) Algorithm Description:* Let $\tilde{V}$ be the set of all real nodes and virtual nodes (line 1). The algorithm first initializes $d[\tilde{v}]$ and $\tilde{d}[\tilde{v}]$ for each node $\tilde{v}$ in $\tilde{V}$ (line 2). Here $d[\tilde{v}]$ denote the shortest $s$-$\tilde{v}$ path length, and $\tilde{d}[\tilde{v}]$ denotes the $s$-$\tilde{v}$ path length that has been found so far. All nodes are classified into two sets $S$ and $\tilde{V} \setminus S$, where $S$ denotes the set of nodes whose shortest path length have not been obtained. Initially, $S$ contains all real nodes and virtual nodes (line 3).

Next, in each round, the algorithm selects the node (say $\tilde{v}$) with the minimum $\tilde{d}[\tilde{v}]$ among the nodes in $S$ (line 5). Based on the property of Dijkstra's algorithm, $d[\tilde{v}]$ is equal to $\tilde{d}[\tilde{v}]$, and so $\tilde{v}$ will be removed from $S$ according to the definition of $S$ (line 6). Let $v$ and $v'$ be the corresponding real node and virtual node of node $\tilde{v}$ (line 7). If $\tilde{v}$ is the first virtual node to be selected, $v'$ (i.e., $\tilde{v}$) has the smallest $d[v']$ among all virtual nodes based on the property of Dijkstra's algorithm. Then, the candidate node is the real node $v$, and the path from the query source $s$ to node $v$ can be constructed by traversing $n[v]$ (line 8). Here, $n[v]$ denotes the predecessor of $v$ on the shortest $s$-$v$ path. If node $\tilde{v}$ is a real node, $n[\cdot]$ will be updated along with $\tilde{d}[\cdot]$ for all $\tilde{v}$'s neighbors (lines 9-12), and the aforementioned procedure will be repeated until the candidate node is selected.

*2) Algorithm Analysis:* Now we calculate the computational complexity. Let $G(V, E)$ be the network graph. After adding virtual nodes and virtual edges, the new weighted graph has $2|V|$ nodes and $|V| + |E|$ edges. The complexity analysis is similar to that of Dijkstra's algorithm. If our algorithm is implemented using a heap-based priority queue, the overall time complexity is $O((|V| + |E|) \log 2|V|)$. Since $|E| > |V|$ ($G$ is connected), the overall time complexity is also equal to $O(|E| \log |V|)$.

## *B. Distributed Approach under Deterministic Expertise Model (DAD)*

CAD assumes the query source has up-to-date information (expertise profile, queuing delay, communication link delay) of every other node, which is impossible due to the communication delay. Thus, we propose a Distributed Approach under

**Algorithm 1** CAD

```
Input: node $s$, $T_v$, $T_v^q$ for $\forall v \in V$, and $c_{u,v}$ for $\forall (u,v) \in E$
Output: the candidate node and the path from node $s$ to that node
 1: $\tilde{V} \leftarrow$ the set of real nodes and virtual nodes
 2: $d[\tilde{v}] \leftarrow \infty$ for $\forall \tilde{v} \in \tilde{V}$, and $\tilde{d}[s] \leftarrow 0$
 3: $S \leftarrow \tilde{V}$
 4: while $S \neq \Phi$ do
 5:     $\tilde{v} \leftarrow$ the node with the minimum $\tilde{d}[\tilde{v}]$ among the nodes in $S$
 6:     $d[\tilde{v}] \leftarrow \tilde{d}[\tilde{v}]$, $S \leftarrow S \setminus \{\tilde{v}\}$
 7:     $v$, $v' \leftarrow$ the real node and the virtual node of $\tilde{v}$, respectively
 8:     If $\tilde{v}$ is $v'$, return the original node $v$ and the $s$-$v$ path constructed by
        traversing $n[v]$
 9:     If $\tilde{d}[v'] > d[v] + T_v + T_v^q$, then $\tilde{d}[v'] \leftarrow d[v] + T_v + T_v^q$
10:     for each real node $u$ which has an edge with $v$ do
11:         If $\tilde{d}[u] > d[v] + 2c_{u,v}$, then $\tilde{d}[u] \leftarrow d[v] + 2c_{u,v}$, and $n[u] \leftarrow v$
12:     end for
13: end while
```

the Deterministic expertise model (DAD) which only requires limited scope of information dissemination. The disseminated information is collected by each node, and further used to calculate the following utility metric to determine where to forward the query.

*Definition 2:* ***Utility***
The utility value of node $v$ for query $q$ is defined as

$$U_v^q = \min_{u \in \mathcal{N}_v} \{2d_{u,v} + T_u + T_u^q\} \tag{6}$$

where $\mathcal{N}_v$ is the set of nodes whose information is used to calculate $U_v^q$. $d_{u,v}$ is the minimum communication delay between node $u$ and node $v$. It is modeled by the length of the shortest $u$-$v$ path in a weighted graph where the link weight is the communication link delay, and it can be calculated using Dijkstra's algorithm. Note that $v \in \mathcal{N}_v$ and $d_{v,v} = 0$.

$U_v^q$ indicates the estimated minimum total delay for responding query $q$, if it is forwarded to node $u$ and processed by the nodes in $\mathcal{N}_v$. The calculation of $U_v^q$ depends on the scope of $\mathcal{N}_v$, which is generally defined as the $r$-hop neighborhood of node $v$ ($r \geq 1$).

Let $B_{u,v}^q$ denote the benefit for node $u$ to forward query $q$ to node $v$. It is defined as the amount of decrease in the total delay if query $q$ is processed by node $v$ instead of node $u$, and can be calculated as follows:

$$B_{u,v}^q = (T_u + T_u^q) - (2c_{u,v} + U_v^q) \tag{7}$$

Generally speaking, each query is greedily forwarded to the nodes which bring maximum benefit in reducing the query delay, and will be processed by the node which does not have any neighbor to bring any benefit. Suppose query $q$ is received/generated by node $u$. Node $u$ first calculates the benefit value of each neighbor. Let $v$ be the node with the maximum $B_{u,v}^q$ in node $u$'s one-hop neighborhood. If $B_{u,v}^q < 0$, node $u$ will process the query by itself. Otherwise, node $u$ will forward the query to node $v$. The aforementioned procedure is repeated until the query is finally processed by some node. The formal description of the algorithm is described in Algorithm 2.

**Information Dissemination:**

Here information dissemination is needed so that each node can calculate the utility values of all nodes in its one-hop

**Algorithm 2** DAD

```
Input: query $q$, node $u$, information disseminated from other nodes
1: $\mathcal{C}_u^q \leftarrow$ the set of $u$'s one-hop neighbors
2: Calculate $B_{u,v}^q$ for each $v$ in $\mathcal{C}_u^q$
3: Sort $\mathcal{C}_u^q$ in a descending order of benefit value
4: $v \leftarrow$ the node in $\mathcal{C}_u^q$ which has the largest benefit
5: if $B_{u,v}^q < 0$ then
6:     Process query $q$
7: else
8:     Forward query $q$ to $v$
9: end if
```

neighborhood. If the calculation of utility value is based on the $r$-hop neighborhood, node $u$ has to maintain the information of all nodes in the $r+1$-hop neighborhood. Thus, each node has to disseminate its information (expertise profile, queuing delay, communication link delay) to all nodes in its $r+1$-hop neighborhood. The details are described as follows.

To obtain the expertise profile, each node estimates the average time to process a query belonging to each domain (as aforementioned in Section III-A2). The communication link delay can be modeled by the average pairwise node inter-contact time. Since the expertise profile and the communication link delay is relatively stable over a long period of time, they only need to be disseminated once after a warm-up period. For the queuing delay, whenever a query is inserted into the queue, the node should re-estimate the queuing delay and re-disseminate it to all nodes in its $r+1$-hop neighborhood. Due to limited scope of information dissemination, there will not be heavy control traffic regarding the dissemination of queuing delay. The information dissemination also takes less time, and hence it is more likely that other nodes can obtain up-to-date information.

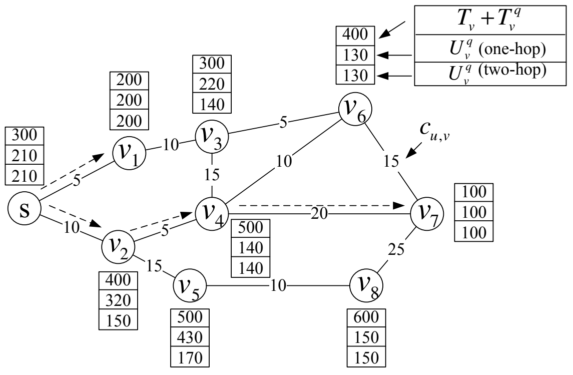


Fig. 3. Illustration of DAD

We use Figure 3 to illustrate how DAD performs with different scope of information dissemination. Suppose a query $q$ is generated by node $s$. We first show the case that the calculation of utility value is based on the one-hop neighborhood. For example, node $v_1$ has $s$ and $v_3$ in its one-hop neighborhood, so $v_1$'s utility value is the minimum of $T_{v_1}+T_{v_1}^q$ (200), $2d_{v_1,s}+T_s+T_s^q$ (310) and $2d_{v_1,v_3}+T_{v_3}+T_{v_3}^q$ (320). Here $T_{v_1}+T_{v_1}^q = 200$, $T_s+T_s^q = 300$, $T_{v_3}+T_{v_3}^q = 300$, $d_{v_1,s} = c_{v_1,s} = 5$, and $d_{v_1,v_3} = c_{v_1,v_3} = 10$ as shown in the figure. Thus, $v_1$'s utility value $U_{v_1}^q$ is 200. Then, the benefit for node $s$ to forward query $q$ to node $v_1$ is $(T_s+T_s^q)-(2c_{s,v_1}+U_{v_1}^q)$, which is equal to 90. Following the same procedure, node $s$ can also obtain the benefit value of node $v_2$, which is equal to $-40$. Thus, the query will be forwarded to node $v_1$ since it has the highest positive benefit value among the nodes in node $s$'s one-hop neighborhood. Then, $v_1$ receives the query and will calculate the benefit value of the neighbors from its own perspective. For example, $B_{v_1,s}^q$ is $(T_{v_1}+T_{v_1}^q)-(2c_{v_1,s}+U_s^q)$, which is equal to $-20$, and $B_{v_1,v_3}^q$ is equal to $-40$. Since no neighbor has positive benefit value, node $v_1$ will process the query. The overall query delay is the sum of $v_1$'s node queuing delay, $v_1$'s processing delay, and twice of the communication delay of link $(s,v_1)$, which is equal to 210.

However, the overall query delay can be reduced if two-hop neighbors are used to calculate the utility metric. In this example, some utility values are reduced, and then the benefit value for a node to forward the query to another node will be updated. Now the query will be forwarded along path $s$-$v_2$-$v_4$-$v_7$ to node $v_7$ for processing, which leads to the overall query delay of 170. Although increasing the scope increases the effectiveness of query forwarding, it also increases the cost of information dissemination. We will further investigate it through extensive simulations.

## V. Query Forwarding Approaches under Probabilistic Expertise Model

In this section, we first propose a Centralized Approach under the Probabilistic expertise model *(CAP)* and then propose a Distributed Approach under the Probabilistic expertise model *(DAP)*.

### A. Centralized Approach under Probabilistic Expertise Model (CAP)

Under the probabilistic expertise model, we cannot assign each query to only the node which has the minimum total delay for processing the query. If the node fails to respond it, the query has to be re-assigned to some other node, which may significantly increase the overall query delay. Thus, we should assign each query to multiple nodes to reduce the failure probability.

In CAP, the source of query $q$ first uses Dijkstra's algorithm to calculate $D_v^q$ for each node $v$. Then, the nodes are sorted in an ascending order of $D_v^q/p_v^q$. The query source will select the first $\alpha$ nodes from the sorted node list, and use source routing to multicast query $q$ to these nodes along the paths selected by Dijkstra's algorithm. Here $\alpha$ is a parameter to be configured by the query source. For each selected node, it sends back the result to the source node through the reverse path that the query had previously traversed to reach the node. If these $\alpha$ nodes are unable to respond query $q$, the query will be multicasted to the next $\alpha$ nodes. The query source continues this procedure until the query is either responded or has been processed by all nodes in the network. The formal description of the algorithm is described in Algorithm 3.

**Effect of Parameter $\alpha$:**

**Algorithm 3** CAP

```
Input: node $s$, $\alpha$, $T_v$, $T_v^q$, $p_v^q$ for $\forall v \in V$, and $c_{u,v}$ for $\forall (u,v) \in E$
1: Consider node $s$ as the source node and calculate $D_v^q$ for $\forall v \in V$ by
   using Dijkstra's algorithm
2: $S \leftarrow V$
3: Sort $S$ in an ascending order of $D_v^q / p_v^q$ for $\forall v \in S$
4: while query $q$ cannot been responded do
5:     $S_\alpha \leftarrow$ the first $\alpha$ nodes from $S$
6:     $S \leftarrow S \setminus S_\alpha$
7:     Multicast query $q$ to the nodes in $S_\alpha$ along the paths selected by
       Dijkstra's algorithm (line 1)
8:     Wait for the results from the selected nodes
9: end while
```

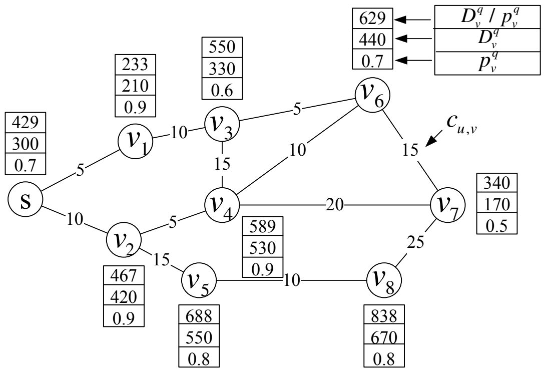


Fig. 4. Illustration of CAP

Now we use an example network to investigate the effect of parameter $\alpha$ on the performance (in terms of the expected query delay) and the overhead (in terms of the expected number of nodes which are assigned a query). In Figure 4, suppose node $s$ generates a query $q$ and assigns it to some nodes for processing using CAP. For each node $v$, $D_v^q/p_v^q$, $D_v^q$ and $p_v^q$ are shown in the figure. The node list in an ascending order of ratio $D_v^q/p_v^q$ is $v_1$ (233), $v_7$ (340), $s$ (429), $v_2$ (467), $v_3$ (550), $v_4$ (589), $v_6$ (629), $v_5$ (688), $v_8$ (838). If $\alpha = 1$, node $s$ will first send the query to $v_1$. If $v_1$ responds the query (the probability is $p_{v_1}^q$), the overall query delay is $D_{v_1}^q$. If $v_1$ cannot respond the query (the probability is $1 - p_{v_1}^q$), node $s$ will send the query to $v_7$ immediately after it obtains the responding result from node $v_1$. If $v_7$ responds the query (the probability is $(1 - p_{v_1}^q)p_{v_7}^q$), the overall query delay is $D_{v_1}^q + D_{v_7}^q$. To summarize, the expected query delay can be calculated as $D_{v_1}^q + (1 - p_{v_1}^q)(D_{v_7}^q + (1 - p_{v_7}^q)\cdots)$, and the expected number of assigned nodes can be calculated as $1 + (1 - p_{v_1}^q)(1 + (1 - p_{v_7}^q)\cdots)$.

Similarly, We can calculate the expected query delay and the expected number of assigned nodes when $\alpha$ is any value within the range $[1, 9]$. The results are shown in Figure 5.

As shown in Figure 5, as $\alpha$ increases, the expected query delay decreases, while the expected number of assigned nodes increases. When a query is assigned to $\alpha$ nodes at once and can be responded by one of them, it is unnecessary to select the other $\alpha - 1$ nodes. Thus, increasing $\alpha$ generally increases the number of unnecessary nodes, which leads to an increases of expected number of assigned nodes. On the other hand, increasing $\alpha$ allows the query to be processed by more nodes at one time, which reduces the overall query delay. As $\alpha$

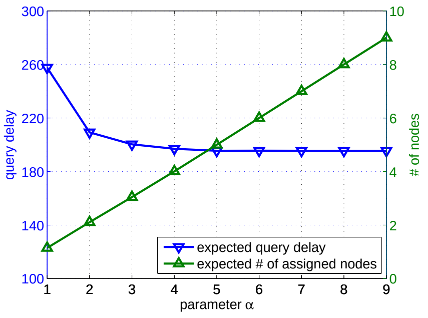


Fig. 5. Effect of $\alpha$ on CAP

increases, more lower-ranked nodes in the sorted node list are selected for processing the query, and the performance improvement (in terms of expected query delay) is much smaller compared with the increase of overhead (in terms of expected number of assigned nodes) by further increasing $\alpha$, as shown in Figure 5. This overhead may affect the performance under the scenario of multiple queries. That is, if the network has few nodes or the query generation interval is very short, assigning a query to many nodes will significantly increase the node queuing delay for other queries. Thus, it is hard to configure $\alpha$ appropriately in order to achieve good performance at the cost of minimal overhead. We propose the following distributed approach to achieve cost effectiveness.

## *B. Distributed Approach under Probabilistic Expertise Model (DAP)*

Since it is impossible to achieve CAP considering the communication delay, we propose a Distributed Approach under Probabilistic Expertise Model (DAP) which only requires limited scope of information dissemination. The disseminated information is collected by each node to calculate the following revised utility metric to guide query multicasting. A node that can process the query with small delay and large responding probability should have better utility value. To achieve cost effectiveness, a query should only be multicasted to a number of nodes which are just enough to respond the query (e.g., the overall responding probability is about 80%). The detailed description of DAP will be given after introducing the following revised utility metric.

*Definition 3:* ***Utility***

The utility value of node $v$ for query $q$ is defined as

$$U_v^q = \min_{u \in \mathcal{N}_v} \{(2d_{u,v} + T_u + T_u^q)/p_u^q\} \tag{8}$$

Similar to DAD, various scopes ($\mathcal{N}_v$) of information are used for calculating the utility metric. Increasing the scope increases the effectiveness of query multicasting (forwarding), but at the cost of information dissemination.

Let $B_{u,v}^q$ denote the benefit for node $u$ to forward query $q$ to node $v$. It can be calculated as follows:

$$B_{u,v}^q = (T_u + T_u^q)/p_u^q - (2c_{u,v} + U_v^q) \tag{9}$$

Suppose query $q$ is received/generated by node $v$. First of all, node $v$ finds out node $v$'s neighbors which have not processed the query (based on some acknowledgement mechanism to be described later). These nodes will be sorted in a descending order of benefit value, and query $q$ will be multicasted to the first $\alpha$ nodes which have positive benefit values. Note that if the query cannot be multicasted to enough appropriate nodes, node $v$ may also process the query.

Generally speaking, the query should be processed by more than one node at once to reduce the failure probability, so we should have $\alpha \geq 2$. However, this may make query $q$ disseminated to an exponential number of nodes, which significantly increases the queuing delay of other queries. Thus, we also estimate the overall responding probability $P_v^q$ that query $q$ has been responded when it is received by node $v$. Note that node $v$ will process the query without further multicasting if (i) $P_v^q \geq \beta$ ($\beta$ is a pre-defined parameter) and (ii) node $v$ has not processed query $q$. Next we describe the details when $\alpha = 2$.

Suppose a query $q$ is generated by node $s$ ($P_s^q$ is initialized to zero). The query may be multicasted to some other nodes for processing. For some node $w$ which generates/receives query $q$, suppose it multicasts the query to nodes $v_1$ and $v_2$. The calculation of $P_{v_1}^q$ ($P_{v_2}^q$) should consider the probability that query $q$ can be responded by the nodes in $v_2$'s ($v_1$'s) neighborhood. For each node $v$, such probability $\tilde{P}_v^q$ is estimated using the responding probability of $v$'s neighbor (say $u$) which has the smallest $(2d_{u,v} + T_u + T_u^q)/p_u^q$ among the nodes in $v$'s neighborhood. Here we assume if node $v$ receives query $q$, it is most likely to forward the query to such node $u$ for processing. Then, $P_{v_1}^q$ and $P_{v_2}^q$ will be calculated as:

$$P_{v_1}^q = 1 - (1 - \tilde{P}_{v_2}^q)(1 - P_w^q) \tag{10}$$

$$P_{v_2}^q = 1 - (1 - \tilde{P}_{v_1}^q)(1 - P_w^q) \tag{11}$$

When node $w$ forwards query $q$ to node $v_1$ ($v_2$), it also appends the value $P_{v_1}^q$ ($P_{v_2}^q$) to the query. Suppose node $v_1$ receives query $q$. If $P_{v_1}^q < \beta$, node $v_1$ will multicast the query to some of its neighbors following the aforementioned procedure. Otherwise, node $v_1$ will process query $q$ without further multicasting if it has not processed the query before.

There are some other cases when the query cannot be multicasted to enough appropriate nodes. For example, it is possible that only some node $v$ can be selected by node $w$. Then, if node $w$ also processes the query, $P_v^q$ is calculated as:

$$P_v^q = 1 - (1 - \tilde{P}_w^q)(1 - P_w^q) \tag{12}$$

Otherwise, $P_v^q$ is equal to $P_w^q$. Note that node $w$ will simply delete the query if (i) node $w$ has processed query $q$ and (ii) query $q$ cannot be forwarded to any appropriate neighbor.

The aforementioned procedures can be generalized for any $\alpha \geq 1$, which are described in Algorithm 4.

**Query Re-multicast and Acknowledgement:** We further address the following two problems: (i) a query cannot be responded by all the nodes that receive the query (ii) a query has been responded by some node but is still processed by other nodes, which is a waste of network resources. To address problem (i), if a node $v$ fails to respond a query $q$, it will set $P_v^q$ to zero and re-multicast query $q$ following Algorithm 4. The candidate nodes will be selected from the neighbors who

**Algorithm 4** DAP

```
Input: parameters $\alpha$, $\beta$, query $q$, node $u$, $P_u^q$, information disseminated from other nodes
 1: If $P_u^q \geq \beta$, process query $q$ and return
 2: $\mathcal{C}_u^q \leftarrow$ the set of $u$'s one-hop neighbors which have not processed $q$
 3: Calculate $B_{u,v}^q$ for each $v$ in $\mathcal{C}_u^q$
 4: Sort $\mathcal{C}_u^q$ in a descending order of benefit value
 5: $\mathcal{S}_u^q \leftarrow$ the set of the first $\alpha$ nodes in $\mathcal{C}_u^q$ which have positive benefit
 6: if $|\mathcal{S}_u^q| < b$ and node $u$ has not processed query $q$ then
 7:     $\mathcal{S}_u^q \leftarrow \mathcal{S}_u^q \bigcup \{u\}$
 8: end if
 9: If $\mathcal{S}_u^q$ is empty, delete query $q$ and return
10: Calculate $\tilde{P}_v^q$ for each $v$ in $\mathcal{S}_u^q$
11: for each $v$ in $\mathcal{S}_u^q$ do
12:     $P_v^q \leftarrow 1 - (1 - P_u^q) \prod_{w \in \mathcal{S}_u^q \setminus \{v\}} (1 - \tilde{P}_w^q)$
13:     If $v$ and $u$ are the same node, process query $q$; otherwise, forward message ($q$,$P_v^q$) to node $v$.
14: end for
```

have not processed query $q$. To achieve this, each node is required to broadcast acknowledgement of whether a query is responded or not after processing it. Then, if node $v$ has not received acknowledgement from a neighbor $u$, node $u$ is assumed to have not processed query $q$. The acknowledgement also addresses problem (ii). That is, if node $v$ knows node $u$ has responded query $q$, it will remove the query from its queue to avoid processing it later.

## VI. Performance Evaluations

In this section, we evaluate the performance of our approaches on both synthetic and realistic traces.

### A. Approaches for Comparison

We evaluate the performance of the four proposed query forwarding approaches: CAD, DAD, CAP, and DAP, and compare them with the following approaches.

**Flooding (Flood):** When a query $q$ is received/generated by some node $v$, $v$ processes $q$ and forwards it to all neighbors. If some node $u$ responds $q$, it will broadcast acknowledgement to all other nodes in the network so that they can remove $q$ from their queues.

**Stochastic Greedy Routing (SGR) [17]:** SGR captures the behavior of humans in forwarding queries. The intuition is that, a query should be forwarded to a node with the following conditions: (i) it has closer expertise to that required by the query, (ii) it has higher node degree, assuming that a better-connected neighbor is more likely to route the query along a shorter path to the node with closer expertise. Specifically, when a node receives a query $q$, it may process the query or forward it to one of its neighbors. The probability of query $q$ to be forwarded to (processed by) node $v$ is proportional to $p_v^q k_v / T_v^q$. Here $k_v$ denotes $v$'s node degree.

Note that we do not implement existing data dissemination schemes [6], [7], [8] for comparison since they do not consider processing delay and cannot fit into our problem formulation. For example, there is a time constraint for each data item in the user-centric data dissemination scheme [6], whereas our problem does not have such constraint but instead aims to minimize the overall query delay.

## B. Synthetic Traces

*1) Simulation Setup:* In our simulations, we randomly generate a mobile opportunistic network consisting of a number of nodes. The pairwise node inter-contact time is randomly generated within the range of $[10, 100]$. There are 10 expertise domains. For the expertise profile of each node $v$, $t_{vi}$ is randomly generated within the range of $[100, 1000]$, and $a_{vi}$ is randomly generated within the range of $[0, 1]$ ($i = 1, 2, \ldots, 10$). We generate a sequence of $10,000$ queries of different expertise requirements. The source of each query is randomly selected. The average query delay will be evaluated under various settings of query generation interval. In all simulations, the first half of the trace is used for warmup to collect necessary network information. All the data and queries are generated during the second half of the trace.

*2) Comparison with Flood and SGR:* Figure 6 and Figure 7 compare the approaches under the deterministic and probabilistic expertise models, respectively.

Figures 6(a), 6(b) (Figures 7(a), 7(b)) compare CAD, DAD (CAP, DAP) with Flood and SGR in a network of 50 nodes, as the query generation interval change. In DAD (DAP), the scope of information dissemination is two-hop. In CAP and DAP, $\alpha = 2$, and $\beta = 0.8$. For all approaches, increasing the query generation interval decreases the average query delay, since the queries are generally buffered for less time before being processed.

In Figure 6(a) (Figure 7(a)), CAD (CAP) generally outperforms DAD (DAP), because CAD (CAP) allows the query source to have up-to-date information of all nodes in the network. For example, when the query generation interval is 20, CAD (CAP) has 9% (26%) lower average query delay than DAD (DAP). When the query generation interval exceeds 100, CAP underperforms DAP since the configuration of parameter $\alpha$ becomes inappropriate for CAP as the query generation interval increases.

In Figure 6(b) (Figure 7(b)), DAD (DAP) outperforms SGR and Flood since the latter two approaches do not consider various kinds of delay information (e.g., communication link delay and node queuing delay). DAD (DAP) has 57%-59% (45%-73%) lower average query delay than SGR, and 96%-97% (93%-96%) lower average query delay than Flood. Flood performs the worst since the node queuing delay is significantly increased.

Figure 6(c) (Figure 7(c)) shows how the performance of DAD (DAP) is affected by the scope of information dissemination when the query generation interval is 20. Here we measure the average query delay and the total number of update messages which are used to disseminate the up-to-date information about node queuing delay.

For DAP (Figure 7(c)), as the number of hops increases from 1 to 3, the average query delay decreases by 34%, while the total number of update messages increases by 212%. As aforementioned in Section IV-B, increasing the scope generally increases the effectiveness of query forwarding. When the number of hops further increases to 4, the average query delay slightly increases. Increasing the scope leads to longer information dissemination time, and so it becomes less likely to obtain up-to-date information (e.g., the node queuing delay may be changed during the information dissemination time). Thus, further increasing the scope reduces the effectiveness of query forwarding. For DAD, the average query delay reaches minimum when the information is only disseminated within one hop. If the scope further increases, the average query delay increases first, and then stays almost flat, which is shown in Figure 6(c).

When the number of hops is more than 4, it is likely that the information dissemination can cover the entire network due to limited network scale. Thus, further increasing the number of hops cannot increase the total number of update messages too much, or significantly change the average query delay.

## C. Realistic Traces

*1) Evaluation Setup:* We further evaluate the performance of our approaches under the probabilistic expertise model using some real-world trace. Here we do not use realistic DTN traces since they do not contain information about node expertise. Our trace is obtained from IBM's IT service department, where service agents collaborate to solve trouble tickets submitted by customers. There are a total of 180 agents and 78,000 tickets which are generated within one month. The tickets are classified into 11 topics, and the topic $id$ reflects the popularity; i.e., a ticket of smaller $id$ is more popular. We assume each topic represents a distinct expertise domain, and use the derived statistics to perform simulations as follows.

We model this network by a graph in which each node represents an agent. If one agent routes a ticket to another agent, an edge is added between them. We randomly generate the pairwise node inter-contact time and study its effect on the performance. We aggregate the tickets by topic and calculate the mean processing time and the probability that a query can be responded by a specific node (referred to as the *responding probability*). Each ticket is viewed as a special kind of queries which only require expertise of one domain. In all simulations, the first half of the tickets is used for warmup to collect necessary network information. All the queries are generated using the second half of the tickets.

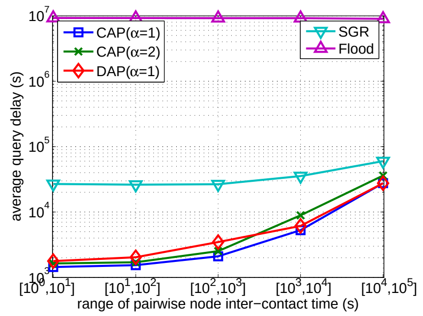
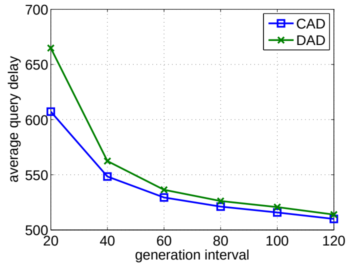
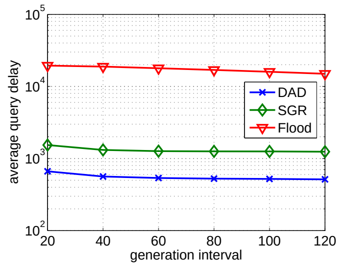


Fig. 8. Results on the realistic trace

*2) Results:* Figure 8 compares the performance of CAP ($\alpha = 1, 2$), DAP ($\alpha = 1$), Flood and SGR as the pairwise node inter-contact time changes. For example, $[10^1, 10^2]$ means that the inter-contact time is randomly generated within $[10^1, 10^2]$s. In DAP, the scope of information dissemination is one-hop.


(a) CAD and DAD


(b) DAD, SGR and Flood

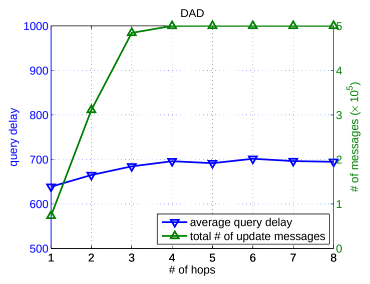
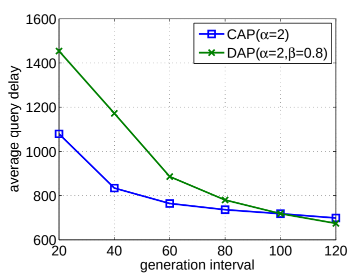
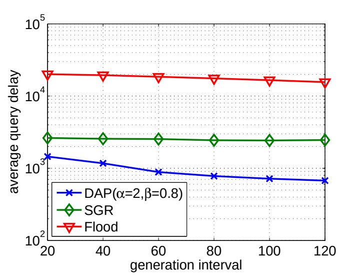

(c) Scope of information dissemination (DAD)

Fig. 6. Deterministic expertise model


(a) CAP and DAP


(b) DAP, SGR and Flood

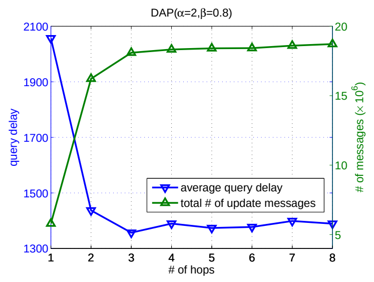

(c) Scope of information dissemination (DAP)

Fig. 7. Probabilistic expertise model

For CAP, DAP and SGR, increasing the pairwise node inter-contact time increases the communication link delay, which increases the overall query delay. Among the three approaches, our approaches have much lower average query delay compared with SGR. For CAP, since there are many queries, increasing $\alpha$ increases node queuing delay, which leads to worse performance. Thus, CAP ($\alpha = 1$) performs better than CAP ($\alpha = 2$). For the same reason, we should reduce the number of nodes selected for query processing in DAP. We set $\alpha = 1$, so a query is processed by one node at a time. Thus, we do not need to configure $\beta$ to restrict the scope of query multicasting.

For Flood, increasing the pairwise node inter-contact time does not change the overall query delay too much. Since there are many queries, flooding significantly increases the node queuing delay. Thus, the query delay is dominated by the node queuing delay, and does not change too much as the pairwise node inter-contact time increases.

## VII. Conclusions

This paper studied the expertise-based data access problem in content-centric mobile opportunistic networks, where the objective is to minimize the average query delay of all queries considering the node expertise, node queuing delay and communication delay. We designed centralized and distributed routing approaches under both deterministic and probabilistic expertise models. Some of our solutions are based on well-known techniques, but with novel modifications. For example, since the network graph has both edge weight (communication delay) and node weight (node queuing delay and processing delay), we need to first transform the original graph by adding virtual nodes and edges before running Dijkstra's algorithm. Extensive simulations on both synthetic and realistic traces demonstrate that our solutions outperform existing approaches.

## References

[1] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *ACM SIGCOMM, 2003*.
[2] W. Gao, G. Cao, T. L. Porta, and J. Han, "On Exploiting Transient Social Contact Patterns for Data Forwarding in Delay-Tolerant Networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 151–165, 2013.
[3] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation Forwarding," in *ACM MobiHoc, 2008*.
[4] W. Gao, Q. Li, and G. Cao, "Forwarding Redundancy in Opportunistic Mobile Networks: Investigation and Elimination," in *IEEE INFOCOM, 2014*.
[5] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," in *ACM SIGCOMM, 2007*.
[6] W. Gao and G. Cao, "User-centric data dissemination in disruption tolerant networks," in *IEEE INFOCOM, 2011*.
[7] K. C.-J. Lin, C.-W. Chen, and C.-F. Chou, "Preference-Aware Content Dissemination in Opportunistic Mobile Social Networks," in *IEEE INFOCOM, 2012*.
[8] L. Guo, C. Zhang, H. Yue, and Y. Fang, "A Privacy-preserving Social-assisted Mobile Content Dissemination Scheme in DTNs," in *IEEE INFOCOM, 2013*.
[9] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[10] *Named Data Networking*. named-data.net.
[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT, 2009*.
[12] *Content Centric Networking Project*. www.ccnx.org.
[13] G. Alfano, M. Garetto, and E. Leonardi, "Content-centric wireless networks with limited buffers: when mobility hurts," in *IEEE INFOCOM, 2013*.
[14] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," in *ACM SIGCOMM, 2007*.
[15] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, "Efficient Ticket Routing by Resolution Sequence Mining," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2008*.
[16] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis, "Generative models for ticket resolution in expert networks," in *ACM KDD, 2010*.
[17] G. Miao, S. Tao, W. Cheng, R. Moulic, L. E. Moser, D. Lo, and X. Yan, "Understanding task-driven information flow in collaborative networks," in *ACM WWW, 2012*.