Cooperative Caching for Efficient Data Access in Disruption Tolerant Networks

Wei Gao, *Member, IEEE*, Guohong Cao, *Fellow, IEEE*,
Arun Iyengar, *Fellow, IEEE* and Mudhakar Srivatsa, *Member, IEEE*

Abstract—Disruption Tolerant Networks (DTNs) are characterized by low node density, unpredictable node mobility and lack of global network information. Most of current research efforts in DTNs focus on data forwarding, but only limited work has been done on providing efficient data access to mobile users. In this paper, we propose a novel approach to support cooperative caching in DTNs, which enables the sharing and coordination of cached data among multiple nodes and reduces data access delay. Our basic idea is to intentionally cache data at a set of Network Central Locations (NCLs), which can be easily accessed by other nodes in the network. We propose an efficient scheme which ensures appropriate NCL selection based on a probabilistic selection metric and coordinates multiple caching nodes to optimize the tradeoff between data accessibility and caching overhead. Extensive trace-driven simulations show that our approach significantly improves data access performance compared to existing schemes.

Index Terms—Cooperative Caching, Disruption Tolerant Networks, Data Access, Network Central Locations, Cache Replacement

1 Introduction

Disruption Tolerant Networks (DTNs) [14] consist of mobile devices which contact each other opportunistically. Due to the low node density and unpredictable node mobility, only intermittent network connectivity exists in DTNs, and the subsequent difficulty of maintaining end-to-end communication links makes it necessary to use "carry-and-forward" methods for data transmission. Examples of such networks include groups of individuals moving in disaster recovery areas, military battlefields, or urban sensing applications [11]. In such networks, node mobility is exploited to let mobile nodes carry data as relays and forward data opportunistically when contacting others. The key problem is therefore how to determine the appropriate relay selection strategy.

Although forwarding schemes have been proposed in DTNs [4], [1], [13], there is limited research on providing efficient data access to mobile users, despite the importance of data accessibility in many mobile applications. For example, it is desirable that Smartphone users can find interesting digital content from their nearby peers. In Vehicular Ad-hoc Networks (VANETs), the availability of live traffic information will be beneficial for vehicles to avoid traffic delays.

In these applications, data is only requested by mobile users whenever needed, and requesters do not know data locations in advance. The destination of data is hence unknown when data is generated. This communication paradigm differs from publish/ subscribe systems [36], [25], in which data is forwarded by broker nodes to users according to their data subscriptions. Appropriate network design is needed to ensure that data can be promptly accessed by requesters in such cases.

A common technique used to improve data access performance is caching, i.e., to cache data at appropriate network locations based on query history, so that queries in the future can be responded with less delay. Although cooperative caching has been studied for both web-based applications [15] and wireless ad-hoc networks [35], [33], [16], [38] to allow sharing and coordination among multiple caching nodes, it is difficult to be realized in DTNs due to the lack of persistent network connectivity. First, the opportunistic network connectivity complicates the estimation of data transmission delay, and furthermore makes it difficult to determine appropriate caching locations for reducing data access delay. This difficulty is also raised by the incomplete information at individual nodes about query history. Second, due to the uncertainty of data transmission, multiple data copies need to be cached at different locations to ensure data accessibility. The difficulty in coordinating multiple caching nodes makes it hard to optimize the tradeoff between data accessibility and caching overhead.

In this paper, we propose a novel scheme to address the aforementioned challenges and to efficiently support cooperative caching in DTNs. Our basic idea is to intentionally cache data at a set of Network Central Locations (NCLs), each of which corresponds to a group of mobile nodes being easily accessed by other nodes in the network. Each NCL is represented by a central node, which has high popularity in the network and is prioritized for caching data. Due to the limited caching

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

- *Wei Gao is with the Department of Electrical Engineering and Computer Science, University of Tennessee at Knoxville, 1520 Middle Drive, Knoxville, TN 37996. E-mail: weigao@utk.edu.*
- *Guohong Cao is with the Department of Computer Science and Engineering, Pennsylvania State University, Information Science and Technology Building, University Park, PA 16802. E-mail: gcao@cse.psu.edu.*
- *Arun Iyengar and Mudhakar Srivatsa are with the IBM T. J. Watson Research Center, Hawthorne, NY 10532. E-mail: {aruni, msrivats}@us.ibm.com.*

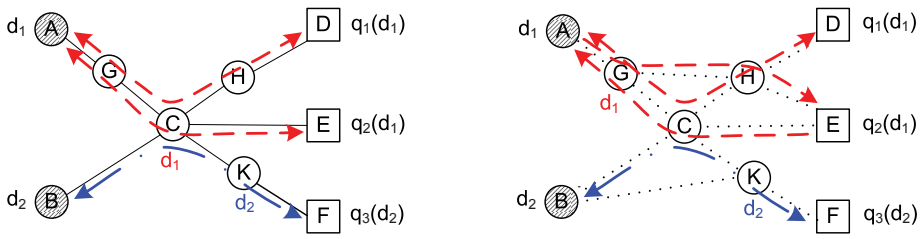


Fig. 1. Caching strategies in different network environments. Data d_1 generated by node A is requested by nodes D and E, and d_2 generated by node B is requested by node F. A solid line in Figure 1(a) between nodes indicates a wireless link, and a dotted line in Figure 1(b) indicates that two nodes opportunistically contact each other.

buffer of central nodes, multiple nodes near a central node may be involved for caching, and we ensure that popular data is always cached nearer to the central nodes via dynamic cache replacement based on query history. Our detailed contributions are listed as follows:

- We develop an efficient approach to NCL selection in DTNs based on a probabilistic selection metric. The selected NCLs achieve high chances for prompt response to user queries with low overhead in network storage and transmission.
- We propose a data access scheme to probabilistically coordinate multiple caching nodes for responding to user queries. We furthermore optimize the tradeoff between data accessibility and caching overhead, to minimize the average number of cached data copies in the network.
- We propose a utility-based cache replacement scheme to dynamically adjust cache locations based on query history, and our scheme achieves good tradeoff between the data accessibility and access delay.

The rest of this paper is organized as follows. In Section 2 we briefly review existing work. Section 3 provides an overview of our approach and highlights our motivation of intentional caching in DTNs. Section 4 describes how to appropriately select NCLs in DTNs. Section 5 describes the details of our proposed caching scheme, and Section 6 proposes load balancing techniques among NCLs. The results of trace-driven performance evaluations are shown in Section 7, and Section 8 concludes the paper.

2 Related Work

Research on data forwarding in DTNs originates from Epidemic routing [34] which floods the entire network. Some later studies focus on proposing efficient relay selection metrics to approach the performance of Epidemic routing with lower forwarding cost, based on prediction of node contacts in the future. Some schemes do such prediction based on their mobility patterns, which are characterized by Kalman filter [8] or semi-Markov chains [37]. In some other schemes, node contact pattern is exploited as abstraction of node mobility pattern for better prediction accuracy [4], [24], based on the experimental [7] and theoretical [5] analysis of the node contact characteristics. The social network properties of node contact patterns, such as the centrality and community structures, have also been also exploited for relay selection in recent social-based data forwarding schemes [9], [22], [20].

The aforementioned metrics for relay selection can be applied to various forwarding strategies, which differ in the number of data copies created in the network. While the most conservative strategy [32] always keeps a single data copy and Spray-and-Wait [31] holds a fixed number of data copies, most schemes dynamically determine the number of data copies. In Compare-and-Forward [12], a relay forwards data to another node whose metric value is higher than itself. Delegation forwarding [13] reduces forwarding cost by only forwarding data to nodes with the highest metric.

Data access in DTNs, on the other hand, can be provided in various ways [28]. Data can be disseminated to appropriate users based on their interest profiles [18]. Publish/ subscribe systems [36], [25] were used for data dissemination, where social community structures are usually exploited to determine broker nodes. In other schemes [24], [2] without brokers, data items are grouped into pre-defined channels, and are disseminated based on users' subscriptions to these channels.

Caching is another way to provide data access. Cooperative caching in wireless ad-hoc networks was studied in [35], in which each node caches pass-by data based on data popularity, so that queries in the future can be responded with less delay. Caching locations are selected incidentally among all the network nodes. Some research efforts [27], [21] have been made for caching in DTNs, but they only improve data accessibility from infrastructure network such as WiFi Access Points (APs) [21] or Internet [27]. Peer-to-peer data sharing and access among mobile users are generally neglected.

Distributed determination of caching policies for minimizing data access delay has been studied in DTNs [29], [23], assuming simpified network conditions. In [29], it is assumed that all the nodes contact each other with the same rate. In [23], users are artificially partitioned into several classes such that users in the same class are iden-

tical. In [19], data is intentionally cached at appropriate network locations with generic data and query models, but these caching locations are determined based on global network knowledge. Comparatively, in this paper we propose to support cooperative caching in a fully distributed manner in DTNs, with heterogeneous node contact patterns and behaviors.

3 Overview

3.1 Motivation

A requester queries the network for data access, and the data source or caching nodes reply to the requester with data after having received the query. The key difference between caching strategies in wireless ad-hoc networks and DTNs is illustrated in Figure 1. Note that each node has limited space for caching. Otherwise, data can be cached everywhere, and it is trivial to design different caching strategies.

The design of caching strategy in wireless ad-hoc networks benefits from the assumption of existing end-to-end paths among mobile nodes, and the path from a requester to the data source remains unchanged during data access in most cases. Such assumption enables any intermediate node on the path to cache the pass-by data. For example, in Figure 1(a), C forwards all the three queries to data sources A and B, and also forwards data d_1 and d_2 to the requesters. In case of limited cache space, C caches the more popular data d_1 based on query history, and similarly data d_2 is cached at node K. In general, any node could cache the pass-by data incidentally.

However, the effectiveness of such an incidental caching strategy is seriously impaired in DTNs, which do not assume any persistent network connectivity. Since data is forwarded via opportunistic contacts, the query and replied data may take different routes, and it is difficult for nodes to collect the information about query history and make caching decision. For example, in Figure 1(b), after having forwarded query q_2 to A, node C loses its connection to G, and cannot cache data d_1 replied to requester E. Node H which forwards the replied data to E does not cache the pass-by data d_1 either, because it did not record query q_2 and considers d_1 less popular. In this case, d_1 will be cached at node G, and hence needs longer time to be replied to the requester.

Our basic solution to improve caching performance in DTNs is to restrain the scope of nodes being involved for caching. Instead of being incidentally cached "anywhere", data is intentionally cached only at specific nodes. These nodes are carefully selected to ensure data accessibility, and constraining the scope of caching locations reduces the complexity of maintaining query history and making caching decision.

3.2 Network Model

Opportunistic contacts in DTNs are described by a network *contact graph* $G(V, E)$, where the stochastic contact process between a node pair $i, j \in V$ is modeled as an edge $e_{ij} \in E$. We assume that node contacts are symmetric; i.e., node j contacts i whenever i contacts j, and the network contact graph is therefore undirected. The characteristics of an edge $e_{ij} \in E$ are determined by the properties of inter-contact time among nodes. Similar to previous work [1], [39], we consider the pairwise node inter-contact time as exponentially distributed. Contacts between nodes i and j then form a Poisson process with contact rate λ_{ij}, which is calculated in real time from the cumulative contacts between nodes i and j since the network starts. In the rest of this paper, we call the node set $\{j|\lambda_{ij} > 0\} \subseteq V$ as the *contacted neighbors* of i.

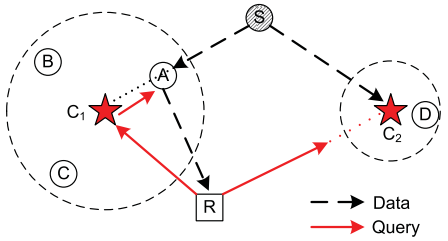


Fig. 2. The big picture of intentional caching

3.3 The Big Picture

We consider a general caching scenario, in which each node may generate data with a globally unique identifier[1] and finite lifetime, and may also request other data by sending queries with a finite time constraint. In practice, such scenario could correspond to various types of disaster environments, which contain mobile users being rescued after hurricane, fire accidents, or earthquake. In these scenarios, the cellular 3G infrastructure may usually be unavailable, or provide too limited bandwidth to transmit data traffic. Instead, mobile users rely on their opportunistic contacts for accessing data, which can be live weather report, traffic condition, or government rescue plan. Similarly, a platoon of soldiers in the military battlefield may lose the satellite connection due to enemy attacks, and have to distribute battle reports via their opportunistic contacts.

In these scenarios, either a data item or a query is described by a set of keywords over a keyword space [18], so that caching nodes can determine the appropriate data that a user is interested in. In these scenarios, data requesters are randomly distributed in the network. We focus on efficiently utilizing the available node buffer to optimize the overall caching performance, which is measured by the successful ratio and delay for mobile users to access different data items.

Our basic idea is to intentionally cache data only at a specific set of NCLs, which can be easily accessed by other nodes in the network. Queries are forwarded to

1. We assign the identifier of each data item as combination of data source's ID and an non-decreasing sequence number, which is maintained by data source and increases whenever a new data item is generated by the source.

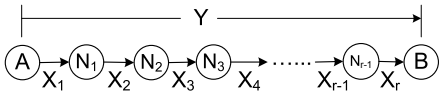


Fig. 3. Opportunistic path

NCLs for data access[2]. The big picture of our proposed scheme is illustrated in Figure 2. Each NCL is represented by a central node, which corresponds to a star in Figure 2. The push and pull caching strategies conjoin at the NCLs. The data source S actively pushes its generated data towards the NCLs, and the central nodes C_1 and C_2 of NCLs are prioritized for caching data. If the buffer of a central node C_1 is full, data is cached at another node A near C_1. Multiple nodes at a NCL may be involved for caching, and a NCL hence corresponds to a connected subgraph of the network contact graph G, as the dashed circles illustrated in Figure 2. Note that NCLs may be overlapping with each other, and a node being involved for caching may belong to multiple NCLs simultaneously. A requester R pulls data by querying NCLs, and data copies from multiple NCLs are returned to ensure prompt data access. Particularly, some NCL such as C_2 may be too far from R to receive the query on time, and does not respond with data. In this case, data accessibility is determined by both node contact frequency and data lifetime.

Nodes in DTNs are well motivated to contribute their local resources for caching data, because the cached data provides prompt data access to the caching nodes themselves. As illustrated by Figure 2, since the central nodes representing NCLs are prioritized for caching data, the closer a requester is to a central node, the sooner its queries are responded by the corresponding NCL. The delay for responding to queries generated from central nodes is, obviously, the shortest.

4 Network Central Locations

In this section, we describe how to select NCLs based on a probabilistic metric evaluating the data transmission delay among nodes in DTNs; we validate the applicability of such metric in practice based on the heterogeneity of node contact pattern in realistic DTN traces. Furthermore, we propose detailed methods for selecting NCLs in practice based on different availability of network information.

4.1 NCL Selection Metric

We first define the multi-hop opportunistic connection on network contact graph $G = (V, E)$.

Definition** 1: **Opportunistic path

A r-hop opportunistic path $P_{AB} = (V_P, E_P)$ between nodes A and B consists of a node set $V_P = \{A, N_1, N_2, ..., N_{r-1}, B\} \subset V$ and an edge set $E_P = \{e_1, e_2, ..., e_r\} \subset E$ with edge weights $\{\lambda_1, \lambda_2, .., \lambda_r\}$. Path weight $p_{AB}(T)$ is the probability that data is opportunistically transmitted from A to B along P_{AB} within time T.

An opportunistic path is illustrated in Figure 3. As described in Section 3.2, the inter-contact time X_k between nodes N_k and N_{k+1} on P_{AB} follows exponential distribution with probability density function (PDF) $p_{X_k}(x) = \lambda_k e^{-\lambda_k x}$. Hence, the time needed to transmit data from A to B is $Y = \sum_{k=1}^{r} X_k$ following a hypoexponential distribution [30], such that

$$p_Y(x) = \sum_{k=1}^{r} C_k^{(r)} p_{X_k}(x), \tag{1}$$

where the coefficients $C_k^{(r)} = \prod_{s=1, s\neq k}^{r} \frac{\lambda_s}{\lambda_s - \lambda_k}$.

From Eq. (1), the path weight is written as

$$p_{AB}(T) = \int_0^T p_Y(x)dx = \sum_{k=1}^{r} C_k^{(r)} \cdot (1 - e^{-\lambda_k T}), \tag{2}$$

and the data transmission delay between two nodes A and B, indicated by the random variable Y, is measured by the weight of the shortest opportunistic path between the two nodes. In practice, mobile nodes maintain the information about shortest opportunistic paths between each other in a distance-vector manner when they come into contact.

The metric C_i for a node i to be selected as a central node to represent a NCL is then defined as follows:

$$C_i = \frac{1}{|V|} \cdot \sum_{j \in V} p_{ij}(T), \tag{3}$$

where we define that $p_{ii}(T) = 0$. This metric indicates the average probability that data can be transmitted from a random node to node i within time T. From Eq. (3), it is obvious that the value of C_i decreases exponentially when T decreases.

TABLE 1
Trace summary

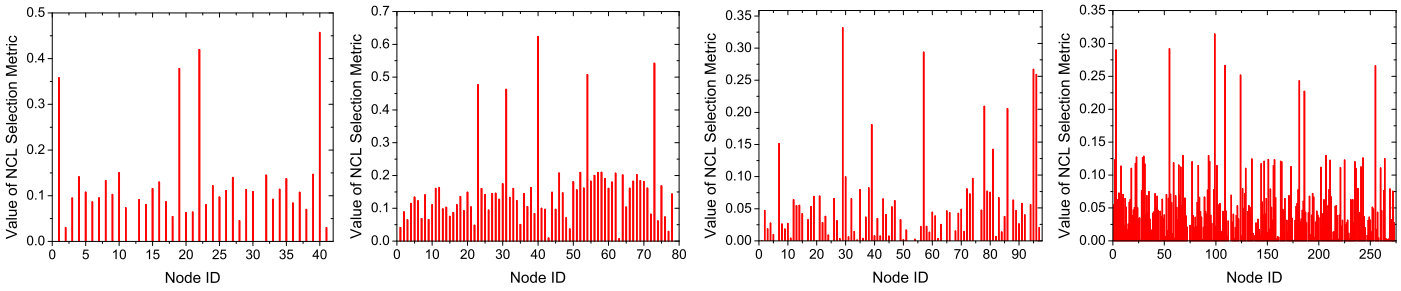
Trace	Infocom05	Infocom06	MIT Reality	UCSD
Network type	Bluetooth	Bluetooth	Bluetooth	WiFi
No. devices	41	78	97	275
No. contacts	22,459	182,951	114,046	123,225
Duration (days)	3	4	246	77
Granularity (secs)	120	120	300	20
Avg. inter-contact time (hours)	3.43	1.83	84.13	47.17

4.2 Trace-based Validation

The practical applicability of the aforementioned NCL selection metric is based on the heterogeneity of node contact patterns, such that nodes in DTNs differ in their popularity and few nodes contact many others frequently. In this section, we validate this applicability using realistic DTN traces.

These traces record contacts among users carrying mobile devices in conference sites and university campuses. The mobile devices, including Mica2 sensors or smartphones, are distributed to users being participated into the experiment. Devices equipped with Bluetooth interface periodically detect their peers nearby, and a

2. Note that our scheme is different from publish/ subscribe system, in which published data is forwarded to subscribers instead of being cached by brokers.



(a) Infocom05 (b) Infocom06 (c) MIT Reality (d) UCSD

Fig. 4. Values of NCL selection metric on realistic DTN traces

contact is recorded when two devices move close to each other. Devices equipped with WiFi interface search for nearby WiFi Access Points (APs) and associate themselves to APs with the best signal strength. A contact is recorded when two devices are associated to the same AP. The detected contacts are recorded in the local storage of mobile devices. After the experiment ends, these devices are called back so that the recorded contacts are processed and analyzed. The traces are summarized in Table 1.

We calculate pairwise node contact rates based on their cumulative contacts during the entire trace. According to Eq. (2), inappropriate values of T make C_i close to 0 or 1. Instead, values of T are adaptively determined in different traces to ensure the differentiation of NCL selection metric values of nodes. T is set as 1 hour for the two Infocom traces, 1 week for the MIT Reality trace, and 3 days for the UCSD trace.

The results in Figure 4 show that the distributions of NCL selection metric values are skewed in all traces, and the metric values of few nodes are much higher than that of others. This difference can be up to tenfold, and suggests that our proposed NCL selection metric efficiently indicates the heterogeneity of node contact pattern. Hence, this metric ensures the selected NCLs can be easily accessed by other nodes.

4.3 Practical NCL Selection

In this section, we propose methods for selecting the required K NCLs in practice based on the NCL selection metric proposed in Section 4.1. We consider K as a predefined parameter determined by the network performance requirements, which will be discussed later in Section 5.5 in more detail.

In general, network information about the pairwise node contact rates and shortest opportunistic paths among mobile nodes are required to calculate the metric values of mobile nodes according to Eq. (3). However, the maintenance of such network information is expensive in DTNs due to the lack of persistent end-to-end network connectivity. As a result, we will first focus on selecting NCLs with the assumption of complete network information from the global perspective. Afterwards, we propose distributed NCL selection methods which efficiently approximate global selection results and can operate on individual nodes in an autonomous manner.

4.3.1 Global Selection

When global network knowledge about the pairwise node contact rates and shortest opportunistic paths among mobile nodes are available, central nodes representing NCLs can be selected sequentially by the network administrator before data access. Let $\mathbb{N}_C$ denote the set of selected central nodes; every time the node in $V \setminus \mathbb{N}_C$ with the highest metric value is selected as the next central node, until the required K central nodes are selected. In particular, we exclude the set $\mathbb{N}_C$ of existing central nodes from calculating C_i in Eq. (3), i.e.,

$$C_i = \frac{1}{|V \setminus \mathbb{N}_C|} \cdot \sum_{j \in V \setminus \mathbb{N}_C} p_{ij}(T). \quad (4)$$

By doing so we ensure that the selected central nodes will not be clustered on the network contact graph. The parameter T used in Eq. (4) is determined by the average node contact frequency. This parameter is generally trace-dependent and was discussed in Section 4.2 for different DTN traces.

A network warm-up period is reserved for nodes to collect information and calculate their pairwise contact rates as described in Section 3.2, and central nodes are selected after the warm-up period ends. Data is unable to be intentionally cached at the NCLs during the warm-up period. Instead, data is incidentally cached by nodes in the network, as described in Section 3.1, for data access. In particular, every requester sends queries directly to the data source, and caches the received data locally for responding other pass-by queries in the future.

After the central nodes representing NCLs are selected, the network administrator is responsible for notifying each node in the network about the information of NCLs via cellular or satellite links. Since each node is only notified about the identifiers of central nodes, this notification is cost-effective without producing noticeable communication overhead, even in cases where the central nodes frequently change. Note that the central nodes are selected due to their popularity in the network, rather than their computation or storage capabilities. Therefore, in general we assume that the central nodes have similar capabilities in computation, data transmission and storage with other nodes in DTNs. Later in

Section 6, we will furthermore study load balancing among central nodes when their local resources are depleted.

4.3.2 Distributed Selection

When global network knowledge is unavailable, a node maintains information about pairwise contact rates and shortest opportunistic paths to other nodes via opportunistic contacts. According to Definition 1 of the opportunistic path, Lemma 1 formally shows that the distributed maintenance of opportunistic paths in DTNs cannot be done in an iterative manner.

***Lemma** 1: There does not exist a function $f(\lambda, T)$, such that for any opportunistic path $P_{AB} = (A, N_1, ..., N_{r-1}, B)$ with edge weights $\{\lambda_1, \lambda_2, ..., \lambda_r\}$,*

$$p_{AB}(T) = p_{AN_{r-1}}(T) \otimes f(\lambda_r, T),$$

where $\otimes$ can be any arbitrary arithmetic operation.

Proof: The difficulty of calculating $p_{AB}(T)$ in an iterative manner mainly comes from the properties of the coefficients $C_k^{(r)}$ in Eq. (2). When a new edge (N_{r-1}, B) with weight λ_r is added into a path AN_{r-1}, such coefficients are modified as

$$C_k^{(r)} = \begin{cases} C_k^{(r-1)} \cdot \frac{\lambda_k}{\lambda_r - \lambda_k}, & k \neq r \\ \prod_{s=1}^{r-1} \frac{\lambda_s}{\lambda_s - \lambda_r}, & k = r \end{cases} \quad (5)$$

We have two observations from Eq. (5). First, each coefficient $C_k^{(r)}$ $(k \neq r)$ is updated by multiplying a distinct value $\frac{\lambda_k}{\lambda_{r+1} - \lambda_k}$. Second, calculation of $C_r^{(r)}$ involves all the edge weights $\lambda_1, ..., \lambda_{r-1}$. Both of them make it impossible to calculate $p_{AB}(T)$ solely from $p_{AN_{r-1}}(T)$ and λ_r. □

Instead, a node i needs to maintain the complete opportunistic paths to other nodes in the network. Initially, each node only has the information about its contacted neighbors. When a node A contacts another node B, they exchange and update their opportunistic path tables. More specifically, for a record of node C in B's table, if C has not been recorded at A, A adds this record into its own table. Otherwise, if the path to C recorded by B has larger weight than that recorded by A, A updates its local record about C.

Being similar with global NCL selection, a network warm-up period is reserved for nodes to exchange and maintain necessary information about opportunistic paths to others. However, a longer warm-up period is needed for distributed NCL selection because multi-hop opportunistic data transmission is required for distributed maintenance of such information.

Afterwards, each node in the network autonomously calculates the value of its NCL selection metric according to Eq. (3) and broadcasts this value to the network. After a pre-defined broadcasting period, a node having received these values then selects the nodes with the K highest metric values as the central nodes representing NCLs.

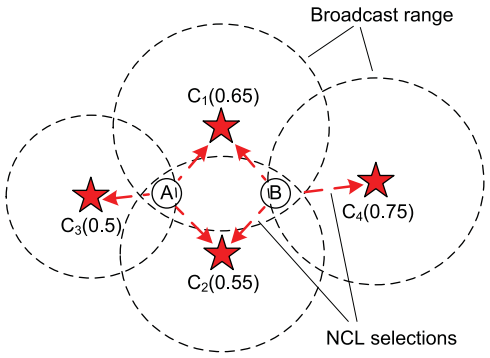


Fig. 5. Inconsistency in distributed NCL selection when $K = 3$

However, due to the uncertainty of opportunistic data transmission in DTNs, the broadcasting range of a particular node may not cover the entire network. Such broadcasting range in DTNs can be formally bounded by the following lemma:

***Lemma** 2: Suppose that node A broadcast its metric value at time t_0 and $\mathbb{S}_A(t_0, t)$ denotes the set of nodes which have received A's information by time $t_0 + t$. We have*

$$\mathbb{P}(|\mathbb{S}_A(t_0, t)| \geq n) \geq (1 - e^{-h_G t})^{n-1}, \quad (6)$$

where

$$h_G = \min_{U \subset V} \frac{\sum_{i \in U, j \in V \setminus U} \lambda_{ij}}{\min(|U|, |V \setminus U|)} \quad (7)$$

is an invariant only depending on the characteristics of the network contact graph.

Proof: Letting T_k be the time by which at least k nodes in the network have received the information of node A, i.e., $T_k = \inf\{t, \text{s.t.} |\mathbb{S}_A(t_0, t)| \geq k\}$, we can easily have

$$\mathbb{P}(T_k - T_{k-1} \leq t) = 1 - e^{-\Lambda t},$$

which means that the random variable $T_k - T_{k-1}$ is exponentially distributed with parameter $\Lambda = \sum_{i \in S, j \in V \setminus S} \lambda_{ij}$, and $S = \mathbb{S}_A(t_0, T_{k-1})$. According to definition of h_G in Eq. (7),

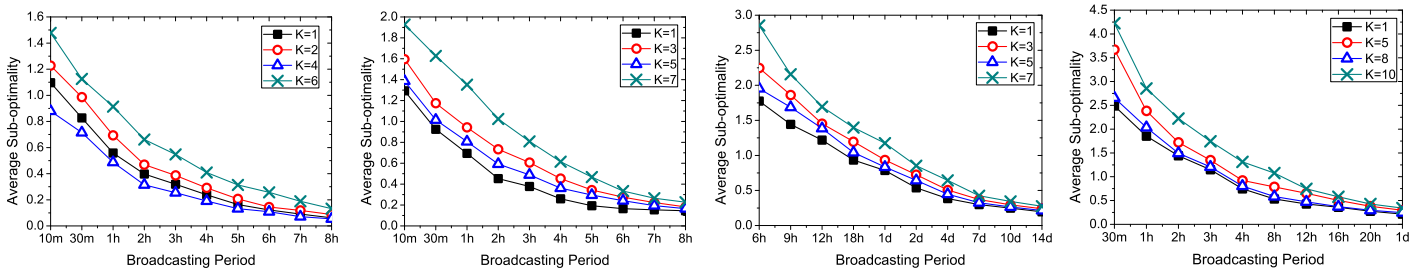
$$\Lambda \geq h_G \cdot \min(|S|, |V \setminus S|) \geq (k-1) h_G,$$

and hence we have

$$\mathbb{P}(|\mathbb{S}_A(t_0, t)| \geq k) \geq 1 - \mathbb{P}(\sum_{j=1}^{k-1} X_j \geq t), \quad (8)$$

where X_j is exponentially distributed with parameter $j h_G$. This lemma is then proved via induction over k based on Eq. (8). □

From Lemma 2, we can see that the metric value of a particular node can only be broadcasted to the entire network after a sufficiently long period of time. In this case, the distributed NCL selections made at individual nodes may be inconsistent. Such inconsistency is illustrated in Figure 5, where the numbers in brackets indicate the values of nodes' NCL selection metric. Due to the limited time for broadcasting, node A is unaware of C_4 which has a high metric value of 0.75, and hence selects C_3 with a lower metric value as the central node when $K = 3$. This sub-optimal selection A made then becomes inconsistent with B's selections.



(a) Infocom05 (b) Infocom06 (c) MIT Reality (d) UCSD

Fig. 6. Sub-optimality of distributed NCL selection on realistic DTN traces

Based on Lemma 2, we have the following theorem which evaluates the occurrence probability of such inconsistency.

***Theorem** 1: After a broadcasting period T, the probability that the NCL selections made by two arbitrary nodes A and B in the network are inconsistent is no larger than $1-(1-2p(1-p))^K$, where K is the pre-defined number of NCLs, and*

$$p = \sum_{n=1}^{N} (1 - e^{-h_G T})^{n-1} \cdot (1 - \frac{n}{N}) \tag{9}$$

Proof: Suppose that $N_1, ..., N_K$ are the central nodes being selected with global network information. For each N_i, if the information of its metric value has been received by n nodes after the broadcasting period T, node A has the probability $1 - n/N$ for not having received such information. Therefore, p in Eq. (9) indicates the probability that the information of N_i has not been received by node A, according to Lemma 2.

It is easy to see that the same probability p also applies to node B. As a result, $1-2p(1-p)$ provides an upper bound on the probability that the NCL selections made by nodes A and B are inconsistent on any N_i for $1 \leq i \leq K$, and this theorem is therefore proved. □

The inconsistency illustrated in Figure 5 can generally be solved by two methods. The first and more straightforward method is to extend the broadcasting period, so that each node is aware of the metric values of all the other nodes in the network. However, this method may be impractical in some mobile applications with strict requirements of timeliness. Another alternative is to opportunistically correct sub-optimal NCL selections when nodes contact each other. More specifically, each node maintains the list of its selected central nodes and their metric values. They exchange such information whenever they contact other and replace the selected central nodes if better ones are found. For example in Figure 5, node A is able to find out that C_4 is a better choice as a central node when it contacts node B.

At last, we evaluate the performance of distributed NCL selection using realistic DTN traces. Due to the aforementioned inconsistency, central nodes selected by individual nodes may be sub-optimal, and we evaluate this sub-optimality at a node i as $\frac{1}{K}\sum_{j=1}^{K}|I_j - j|$, where the j-th central node selected by node i has the I_j-th largest metric value in the network[3]. The average sub-optimality over all the nodes in the network with different broadcasting periods T is shown in Figure 6. In general, Figure 6 shows that the performance of distributed NCL selection is closely related with the length of T. The selected central nodes are far from optimal when T is small, but will be quickly improved when T increases. When T is sufficiently large, the performance of distributed NCL selection closely approximates that of global selection.

Figure 6 shows that the sub-optimality of selected central nodes generally increases with the value of K, which is consistent with our theoretical expectation in Theorem 1. However, by comparing Figure 6 with Figure 4, we notice that this sub-optimality may also be diminished if the value of K is appropriately selected to reflect the heterogeneity of network contact pattern. For example, Figure 4(b) shows that the metric values of 5 nodes are much higher than those of other nodes in the *Infocom06* trace. Correspondingly, the sub-optimality of distributed NCL selection can be reduced as shown in Figure 6(b) when the value of K is changed from 3 to 5. Similar cases are also found in all the other traces.

5 Caching Scheme

In this section, we present our cooperative caching scheme. Our basic idea is to intentionally cache data at a set of NCLs which can be promptly accessed by other nodes. Our scheme consists of the following three components:

1) When a data source generates data, it pushes data to central nodes of NCLs which are prioritized to cache data. One copy of data is cached at each NCL. If the caching buffer of a central node is full, another node near the central node will cache the data. Such decisions are automatically made based on buffer conditions of nodes involved in the pushing process.
2) A requester multicasts a query to central nodes of NCLs to pull data, and a central node forwards the query to the caching nodes. Multiple data copies are returned to the requester, and we optimize

3. We have $I_j = j$ for $\forall j$ if the optimal central nodes are selected.

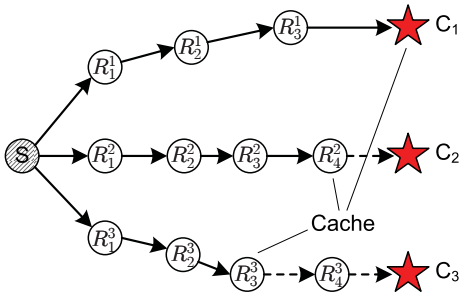


Fig. 7. Determining caching location at NCLs

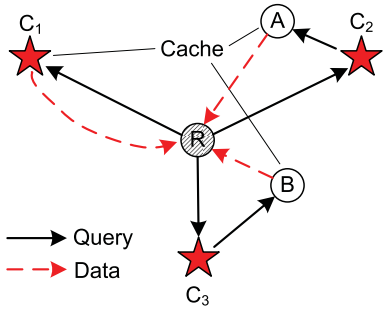


Fig. 8. Pulling data from the NCLs

the tradeoff between data accessibility and transmission overhead by controlling the number of returned data copies.

3) Utility-based cache replacement is conducted whenever two caching nodes contact and ensures that popular data is cached nearer to central nodes. We generally cache more copies of popular data to optimize the cumulative data access delay. We also probabilistically cache less popular data to ensure the overall data accessibility.

5.1 Caching Location

Whenever a node S generates new data, S pushes the data to NCLs by sending a data copy to each central node representing a NCL. We use the opportunistic path weight to the central node as relay selection metric for such data forwarding, and a relay forwards data to another node with a higher metric than itself. This "Compare-and-Forward" strategy has been widely used in the literature [10], [9] for efficient data forwarding. According to Definition 1 on opportunistic path, this strategy probabilistically ensures that each forwarding reduces the remaining delay for data to be delivered to the central node.

For newly generated data, the initial caching locations are automatically determined during the forwarding process based on node buffer conditions. The caching locations are then dynamically adjusted by cache replacement described in Section 5.4 according to query history. In general, data is forwarded to and cached at central nodes. This forwarding process only stops when the caching buffer of the next relay is full[4], and data is cached at the current relay in such cases. In other words, during the data forwarding process towards central nodes, relays carrying data are considered as temporal caching locations of the data.

Such determination of caching location is illustrated in Figure 7, where the solid lines indicate opportunistic contacts used to forward data, and the dashed lines indicate data forwarding stopped by node buffer constraint. Central node C_1 is able to cache data, but data copies to C_2 and C_3 are stopped and cached at relays R_4^2 and R_3^3 respectively, because neither C_2 nor R_4^3 has enough buffer to cache data. Note that the caching location at a NCL may not be the contacted neighbor of a central node, like the case of nodes R_3^3 in Figure 7.

From this strategy, it is easy to see that the set of caching nodes at each NCL forms a connected subgraph of the network contact graph at any time during data access. This property essentially facilitates the delivery of user queries to the caching nodes, which is described in Section 5.2.

5.2 Queries

We assume that any node may request data, and hence data requesters are randomly distributed in the network. A requester multicasts a query with a finite time constraint to all the central nodes to pull data, and existing multicast schemes in DTNs [20] can be exploited for this purpose.

After having received the query, a central node immediately replies to the requester with data if it is cached locally[5]. Otherwise, it broadcasts the query to the nodes nearby. This process is illustrated in Figure 8. While the central node C_1 is able to return the cached data to R immediately, the caching nodes A and B only reply to R after they receive the query from central nodes C_2 and C_3, respectively. The query broadcast finishes when query expires. Each caching node at NCLs maintains up-to-date information about query history, which is used in Section 5.4 for cache replacement.

5.3 Probabilistic Response

As shown in Figure 8, multiple data copies are replied to the requester from NCLs to ensure that the requester receives data before query expires. However, only the first data copy received by the requester is useful, and all the others are essentially useless and waste network resources. The major challenge for solving this problem arises from the intermittent network connectivity in DTNs. First, it is difficult for caching nodes to promptly communicate with each other, and hence the optimal number of data copies returned to the requester cannot be determined in advance. Second, a relay carrying a data copy does not know the locations of other data copies being returned, and therefore cannot determine whether the requester has received data.

4. Since the data is newly generated and has not been requested yet, no cache replacement is necessary at the relay.

5. Particularly, if a caching node is selected as the relay during multicasting of a query, it directly sends the cached data to the requester, without forwarding the query to the central node.

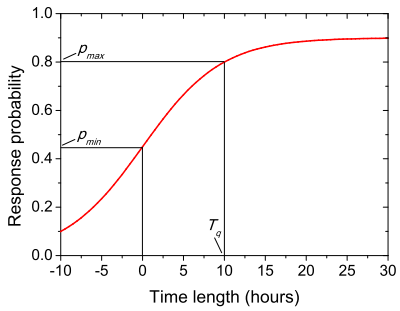


Fig. 9. Probability for deciding data response

In this section, we propose a probabilistic scheme to address these challenges and optimize the tradeoff between data accessibility and transmission overhead. Our basic idea is that, having received the query, a caching node probabilistically decides whether to return the cached data to the requester. Different strategies are used for this decision, according to the availability of network contact information.

We assume that a query is generated with a time constraint T_q, and it takes $t_0 < T_q$ for it to be forwarded from requester R to caching node C. If C knows the information about the shortest opportunistic paths to all the nodes in the network, C can determine whether to reply data to R with the probability $p_{CR}(T_q - t_0)$. According to Eq. (2), $p_{CR}(T_q - t_0)$ indicates the probability that data can be transmitted from C to R within the remaining time $T_q - t_0$ for responding to the query.

Otherwise, C only maintains information about shortest opportunistic paths to central nodes, and it is difficult for C to estimate the data transmission delay to R. Instead, the probability for deciding data response is calculated only based on the remaining time $T_q - t_0$. This probability should be proportional to $T_q - t_0$, and we calculate this probability as a Sigmoid function $p_R(t)$, where $p_R(T_q) = p_{\max} \in (0, 1]$ and $p_R(0) = p_{\min} \in (p_{\max}/2, p_{\max})$. This function is written as

$$p_R(t) = \frac{k_1}{1 + e^{-k_2 \cdot t}}, \tag{10}$$

where $k_1 = 2p_{\min}$, $k_2 = \frac{1}{T_q} \cdot \ln(\frac{p_{\max}}{2p_{\min} - p_{\max}})$. The quantities $p_{\max}$ and $p_{\min}$ in Eq. (10) are user-specified parameters of the maximum and minimum response probabilities. As an example, the sigmoid function with $p_{\min} = 0.45$, $p_{\max} = 0.8$, and $T_q = 10$ hours is shown in Figure 9.

5.4 Cache Replacement

For each data item in the network, the locations where it is cached are dynamically adjusted via cache replacement. This replacement is based on data popularity, and generally places popular data nearer to the central nodes of NCLs. Traditional cache replacement strategies such as LRU, which removes the least-recently-used data from cache when new data is available, are ineffective due to its over-simplistic consideration of data popularity. Greedy-Dual-Size [6] calculates data utility by considering data popularity and size simultaneously, but cannot ensure optimal selection of cached data. We improve previous work by proposing a probabilistic cache replacement strategy, which appropriately selects the data to be cached and heuristically balances between the cumulative data accessibility and access delay.

5.4.1 Data Popularity

The popularity of a data item is probabilistically estimated based on the past k requests to this data during time period $[t_1, t_k]$. We assume that such occurrences of data requests follow a Poisson distribution with the parameter $\lambda_d = k/(t_k - t_1)$, and data popularity is defined as the probability that this data will be requested again in the future before data expires. If data d_i expires at time t_e, its popularity is $w_i = 1 - e^{-\lambda_d \cdot (t_e - t_k)}$. To calculate w_i, a node only needs to recursively maintain two time values about the past occurrences of data requests, and therefore will only incur negligible space overhead.

5.4.2 Basic Strategy

Cache replacement opportunistically occurs whenever two caching nodes A and B contact. The two nodes exchange their cached data to optimize the cumulative data access delay[6]. We collect the cached data at both nodes into a selection pool $\mathbb{S} = \{d_1, ..., d_n\}$, and formulate cache replacement as follows:

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i u_i + \sum_{j=1}^{n} y_j v_j \\ &\text{s.t.} \sum_{i=1}^{n} x_i s_i \le S_A, \sum_{j=1}^{n} y_j s_j \le S_B \\ &x_i + y_i \le 1, \text{ for } \forall i \in [1, n], \end{aligned} \tag{11}$$

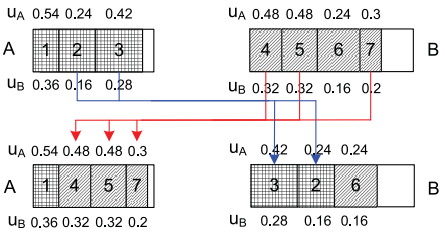
where $x_i, y_i \in [0, 1]$ indicate whether data d_i is cached at node A and B after replacement, respectively. s_i indicates size of data d_i, and S_A and S_B are the buffer sizes of A and B. $u_i = w_i \cdot p_A$ and $v_i = w_i \cdot p_B$ indicate the utility of data d_i at A and B to cumulative caching performance, where w_i is popularity of data d_i; p_A and p_B are the weight of the shortest opportunistic path to the corresponding central node.

This formulation places popular data to caching nodes near the central nodes. It is NP-hard since the standard 0-1 knapsack problem can reduce to this problem. We propose a heuristic to approximate the solution of this problem.

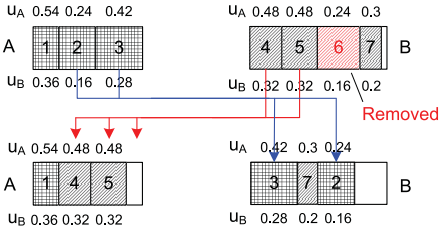
Without loss of generality we assume that $p_A > p_B$, and node A is prioritized to select its data to cache from $\mathbb{S}$ by solving the following problem extracted from Eq. (11):

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i u_i \\ &\text{s.t.} \sum_{i=1}^{n} x_i s_i \le S_A. \end{aligned} \tag{12}$$

6. Since nodes only exchange data when they contact, it is unnecessary for a caching node to actively remove obsolete data from its local cache.



(a) Normal cache replacement



(b) Possible data removal

Fig. 10. Cache replacement

Afterwards, node B selects data to cache from the remaining part of $\mathbb{S}$ by solving a similar problem to Eq. (12). Since S_A and s_i in Eq. (12) are usually integers in numbers of bytes, this problem can be solved in pseudo-polynomial time $O(n{\cdot}S_A)$ using a dynamic programming approach [26].

This replacement process is illustrated in Figure 10, where initially node A caches data d_1, d_2 and d_3, and node B caches data d_4, d_5, d_6 and d_7. The two nodes exchange and replace their cached data upon contact, based on the data utility values listed as u_A and u_B. As shown in Figure 10(a), since $p_A > p_B$, node A generally caches the popular data d_4, d_5 and d_7, and leaves data d_2 and d_3 with lower popularity to node B.

In cases of limited cache space, some cached data with lower popularity may be removed from caching buffer. In Figure 10(b), when the sizes of caching buffer of nodes A and B decrease, A does not have enough buffer to cache data d_7, which is instead cached at node B. Data d_6 with the lowest popularity will then be removed from cache, because neither node A nor B has enough space to cache it.

5.4.3 Probabilistic Data Selection

The aforementioned removal of cached data essentially prioritizes popular data during cache replacement, but may impair the cumulative data accessibility. The major reason is that, according to our network modeling in Section 3.2, the data accessibility does not increase linearly with the number of cached data copies in the network. More specifically, the data accessibility will increase considerably if the number of cached data copies increases from 1 to 2, but the benefit will be much smaller if the number increases from 10 to 11. In such cases, for the example shown in Figure 10(b), caching d_1 at node A may be ineffective, because the popular d_1 may already be cached at many other places in the network. In contrast, removing d_6 out from the cache of node B may greatly impair the accessibility of d_6, because there may be only few cached copies of d_6 due to its lower popularity.

Algorithm 1: Probabilistic Data Selection at node A among the data set $\mathbb{S}$

$i_{\min} = \arg\min_i \{s_i | d_i \in \mathbb{S}, x_i == 0\}$
while $\mathbb{S} \neq \emptyset$ *&&* $S_A > s_{i_{\min}}$ **do**
  $V_{\max}$ = **GetMax**($\mathbb{S}$, S_A)
  $\mathbb{S}' = \mathbb{S}$
  while $\mathbb{S}' \neq \emptyset$ *&&* $V_{\max} > 0$ **do**
    $i_{\max} = \arg\max_i \{u_i | d_i \in \mathbb{S}'\}$
    if **SelectData**($d_{i_{\max}}$)==true *&&* $V_{\max} \geq s_{i_{\max}}$ **then**
      $x_{i_{\max}} = 1$
      $\mathbb{S} = \mathbb{S} \setminus d_{i_{\max}}$
      $S_A = S_A - s_{i_{\max}}$, $V_{\max} = V_{\max} - s_{i_{\max}}$
    $\mathbb{S}' = \mathbb{S}' \setminus d_{i_{\max}}$
  $i_{\min} = \arg\min_i \{s_i | d_i \in \mathbb{S}, x_i == 0\}$

In other words, the basic strategy of cache replacement only optimizes the cumulative data access delay within the local scope of the two caching nodes in contact. Such optimization at the global scope is challenging in DTNs due to the difficulty of maintaining knowledge about the current number of cached data copies in the network, and we instead propose a probabilistic strategy to heuristically control the number of cached data copies at the global scope.

The basic idea is to probabilistically select data to cache when the problem in Eq. (12) is solved by a dynamic programming approach. More specifically, if data d_i is selected by the dynamic programming algorithm, it has probability u_i to be cached at node A. This algorithm is described in detail in Algorithm 1, where **GetMax**($\mathbb{S}, S_A$) calculates the maximal possible value of the items in the knapsack via dynamic programming, and **SelectData**($d_{i_{\max}}$) determines whether to select data $d_{i_{\max}}$ to cache at node A by conducting a Bernoulli experiment with probability $u_{i_{\max}}$. Such probabilistic selection may be iteratively conducted multiple times to ensure that the caching buffer is fully utilized. By proposing this probabilistic strategy, we still prioritize the popular data with higher utility during the caching decision, but also enable the data with less popularity to have non-negligible chance to be cached.

5.5 Discussions

In summary, data access delay of our scheme consists of three parts: i) the time for query to be transmitted from requester to central nodes; ii) the time for central nodes to broadcast query to caching nodes; iii) the time for the cached data to be returned to requester.

Data access delay is closely related to the number (K) of NCLs. When K is small, the average distance from a node to the NCLs is longer, which makes the first and third parts of the delay bigger. Meanwhile, since

the total amount of data being cached in the network is small, data is more likely to be cached near to the central nodes, and the second part of the delay can be short. In contrast, if K is large, the metric values of some central nodes may not be high, and hence caching at the corresponding NCLs may be less efficient. Moreover, when the node buffer constraint is tight, a caching node may be shared by multiple NCLs. The NCLs with lower caching effectiveness may disturb the caching decision of other NCLs and furthermore impair the caching performance.

It is clear that the number (K) of NCLs is vital to the performance of our caching scheme. In Section 7.4, we will experimentally investigate the impact of different values of K on the caching performance in more details.

6 NCL Load Balancing

From the caching scheme proposed in Section 5, we can see that the central nodes play vital roles in cooperative caching in DTNs. First, the central nodes cache the most popular data in the network and respond to the frequent queries for these data. Second, the central nodes are also responsible for broadcasting all the queries they receive to other caching nodes nearby. However, such functionality may quickly consume the local resources of central nodes which include their battery life and local memory. In addition, we would like our caching schemes to be resilient to failures of central nodes. In this section, we focus on addressing this challenge, and propose methods which efficiently migrate the functionality of central nodes to other nodes in cases of failures or resource depletion. In general, the methods we present in this section can be used to adjust the deployment of central nodes at run-time, such as adding or removing central nodes according to up-to-date requirements on caching performance.

6.1 Selecting the New Central Node

When a central node fails or its local resources are depleted, another node is selected as a new central node. Intuitively, the new central node should be the one with the highest NCL selection metric value among the current non-central nodes in the network. However, such selection may degrade the caching performance as illustrated in Figure 11. When the local resources of central node C_1 are depleted, its functionality is taken over by C_3. Since C_3 may be far away from C_1, the queries broadcasted from C_3 may take a long time to reach the caching nodes A, and hence reduce the probability that the requester R receives data from A on time. From Figure 11, it is easy to see that such performance degradation is caused by the existing data being cached at nodes near C_1.

In this case, the distance between the new central node and C_1 should also be taken into account. More specifically, with respect to the original central node j,

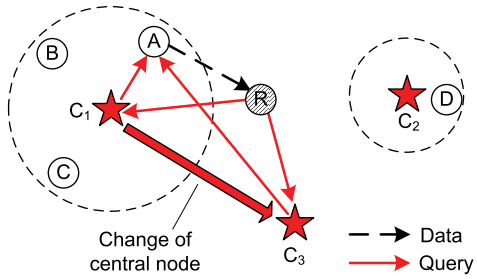


Fig. 11. NCL load balancing

we define the metric C_i^j for a node i to be selected as the new central node as

$$C_i^j = C^i \cdot p_{ij}(T), \tag{13}$$

where C_i is the original NCL selection metric defined in Section 4 and $p_{ij}(T)$ is the weight of the shortest opportunistic path between node i and j defined in Eq. (2).

In practice, an existing central node j is responsible for selecting the new central node i when its local resources are depleted according to the metric defined in Eq. (13); node j also broadcasts a notification to the entire network indicating the new central node. If node j is unable to do so due to sudden failure, another node in contact with j will be responsible for such selection and notification. To realize this, node j designates one of its contacted neighbors with the maximum battery life as its "backup", and synchronizes this backup node with all the information that j has regarding the contact capabilities of other nodes. As a result, when node j suddenly fails due to resource depletion, this backup node will be responsible for selecting the new central node.

6.2 Adjustment of Caching Locations

After a new central node is selected, the data cached at the NCL represented by the original central node needs to be adjusted correspondingly, so as to optimize the caching performance. For example in Figure 11, after the functionality of central node C_1 has been migrated to C_3, the nodes A, B and C near C_1 are not considered as good locations for caching data anymore. Instead, the data cached at these nodes needs to be moved to other nodes near C_3.

This movement is achieved via cache replacement when caching nodes opportunistically contact each other. Each caching node at the original NCL re-calculates the utilities of its cached data items with respect to the newly selected central node. In general, these data utilities will be reduced due to the changes of central nodes, and this reduction moves the cached data to the appropriate caching locations which are nearer to the newly selected central node.

Changes in central nodes and subsequent adjustment of caching locations inevitably affect caching performance as shown in Figure 11. However, this performance degradation will be gradually eliminated over time by the opportunistic cache replacement. In Section 7, we

will furthermore evaluate such impact in practice on realistic DTN traces.

7 Performance Evaluation

We evaluate the performance of our proposed caching scheme by comparing it with the following schemes:

- **No Cache**, where caching is not used for data access and each query is only responded by data source.
- **Random Cache**, in which every requester caches the received data to facilitate data access in the future.
- **CacheData** [35], which is proposed for cooperative caching in wireless ad-hoc networks, and lets each relay in DTNs cache the pass-by data based on their popularity.
- **Bundle Cache** [27], which packs network data as bundles and makes caching decision on pass-by data by considering the node contact pattern in DTNs, so as to minimize the average data access delay.

Cache replacement algorithms are proposed in CacheData and Bundle Cache, and will also be used in our evaluations. For Random Cache, LRU is used for cache replacement. The following metrics are used for evaluations. Each simulation is repeated multiple times with randomly generated data and queries for statistical convergence.

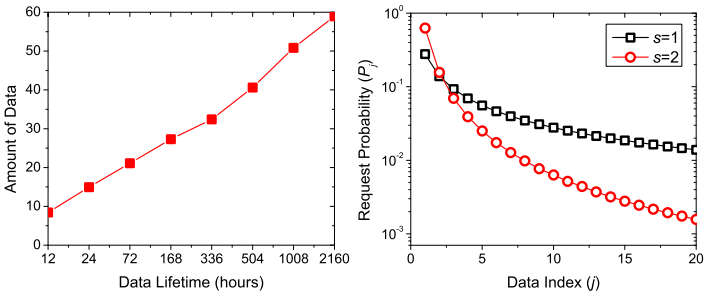
- **Successful ratio**, the ratio of queries being satisfied with the requested data. This ratio evaluates the coverage of data access provided by our proposed caching schemes.
- **Data access delay**, the average delay for getting responses to queries.
- **Caching overhead**, the average number of data copies being cached in the network[7].

7.1 Experiment Setup

Our performance evaluations are performed on the *Infocom06* and *MIT Reality* traces. In all the experiments, central nodes representing NCLs are globally selected before data and queries are generated. The first half of the trace is used as warm-up period for the accumulation of network information and subsequent NCL selection, and all the data and queries are generated during the second half of trace.

7.1.1 Data Generation

Each node periodically checks whether it has generated data which has not expired yet. If not, the node determines whether to generate new data with probability p_G. Each generated data has finite lifetime uniformly distributed in range $[0.5T, 1.5T]$, and the period for data generation decision is also set as T. In our evaluations we fix $p_G = 0.2$, and the amount of data in the network is hence controlled by T, as illustrated in Figure 12(a) for the *MIT Reality* trace. Similarly, data size is uniformly distributed in range $[0.5s_{avg}, 1.5s_{avg}]$, and caching buffers of nodes are uniformly distributed in range $[200\text{Mb}, 600\text{Mb}]$. s_{avg} is adjusted to simulate different node buffer conditions.



(a) Amount of network data (b) Data request probabilities

Fig. 12. Experiment setup

Note that in this section, we compare the performance of our proposed schemes with the existing work. When T is large, indicating long inter-contact time among mobile nodes in the network, our experimental setup increases the data lifetime accordingly. In this way, we ensure non-negligible caching performance in the network and furthermore comprehensive performance comparisons. We could reasonably infer that the comparison results we have in this section will still hold, when the average inter-contact time in the network is reduced and enables efficient access on data with shorter lifetime.

7.1.2 Query Pattern

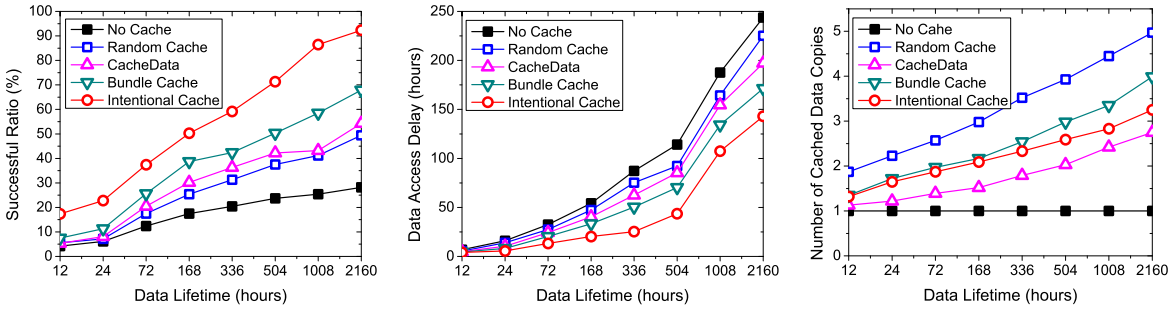
Queries are randomly generated at all nodes, and each query has a finite time constraint $T/2$. In particular, every time $T/2$, each node independently determines whether to generate a query for data j with probability P_j. We assume that query pattern follows a Zipf distribution which has been proved to describe the query pattern of web data access [3]. As a result, letting M be the number of data items in the network, we have $P_j = \frac{1}{j^s} / (\sum_{i=1}^{M} \frac{1}{i^s})$ where s is an exponent parameter. Values of P_j with different s are shown in Figure 12(b).

7.2 Caching Performance

Caching performance of our scheme is evaluated using *MIT Reality* trace. The number (K) of NCLs is set to 8 and query pattern follows a Zipf distribution with $s = 1$. By default, $T = 1$ week and $s_{avg} = 100$ Mb. These two parameters are then adjusted for different performance evaluation purposes.

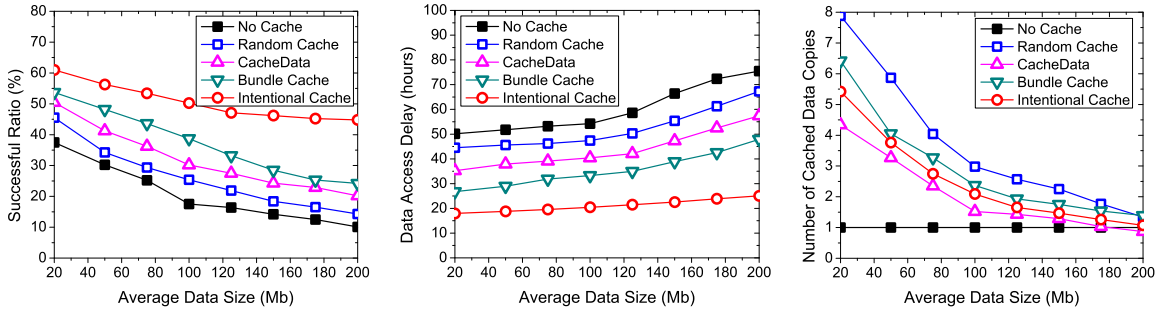
The simulation results with different values of T are shown in Figure 13. The successful ratio of data access is mainly restrained by T itself. When T increases from 12 hours to 3 months, the successful ratio of all schemes is significantly improved, because data has more time to be delivered to requesters before expiration. Since the selected NCLs are efficient in communicating with other nodes, our proposed intentional caching scheme achieves much better successful ratio and delay of

7. We consider the overhead of maintaining node contact information as negligible, because only the pairwise contact rates are maintained for calculating NCL selection metric, as described in Section 4.1.



(a) Successful ratio (b) Data access delay (c) Caching overhead

Fig. 13. Performance of data access with different data lifetime



(a) Successful ratio (b) Data access delay (c) Caching overhead

Fig. 14. Performance of data access with different node buffer conditions

data access. As shown in Figures 13(a) and 13(b), the performance of our scheme is 200% better than that of NoCache, and also exhibits 50% improvement over BundleCache where nodes also incidentally cache pass-by data. Comparatively, RandomCache is ineffective due to the random distribution of requesters in the network, and CacheData is also inappropriate for DTNs due to the difficulty of maintaining query history.

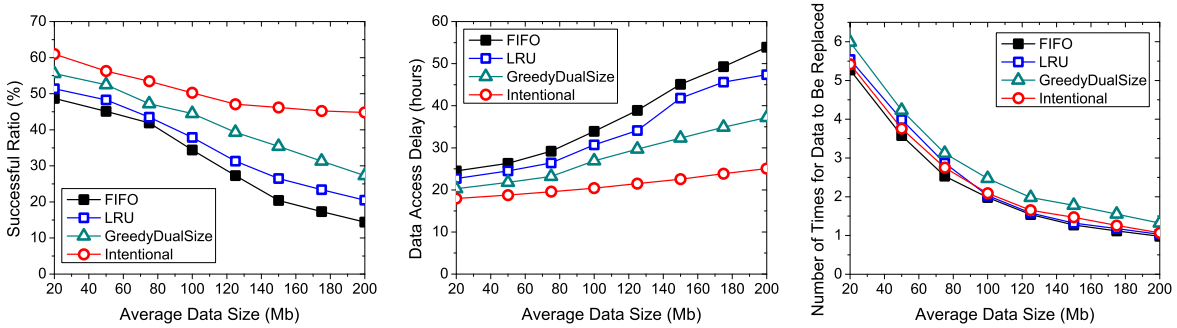
Meanwhile, Figure 13(c) shows that our scheme only requires moderate cache size, which is much lower than that required by RandomCache and BundleCache, especially when T is large. RandomCache consumes the largest caching buffer, such that each data has 5 cached copies when T increases to 3 months. The major reason is that each requester blindly caches any received data until its buffer is filled up. CacheData consumes 30% less buffer than our scheme, but also leaves a lot of data uncached which impairs data access performance. We notice that caching overhead in our scheme also includes the transmission and storage cost when queries and data are transmitted between requesters and caching nodes, and realize that such cost is proportional to data access delay during which data is carried by relays. Hence, the cost-effectiveness of our scheme is also supported by Figure 13(b).

We also evaluated data access performance with different node buffer conditions by adjusting s_{avg}, and the results are shown in Figure 14. When data size becomes larger, less data can be cached as shown in Figure 14(c), and data access performance is hence reduced. In Figures 14(a) and 14(b), when s_{avg} increases from 20Mb to 200Mb, the successful ratio of our scheme decreases from 60% to 45%, and data access delay increases from 18 hours to 25 hours. However, the performances of other schemes even decrease much faster, and the advantage of our scheme becomes even larger when node buffer constraint is tight. This is mainly due to the intelligent cache replacement strategy used in our scheme, which ensures that the most appropriate data is cached.

7.3 Effectiveness of Cache Replacement

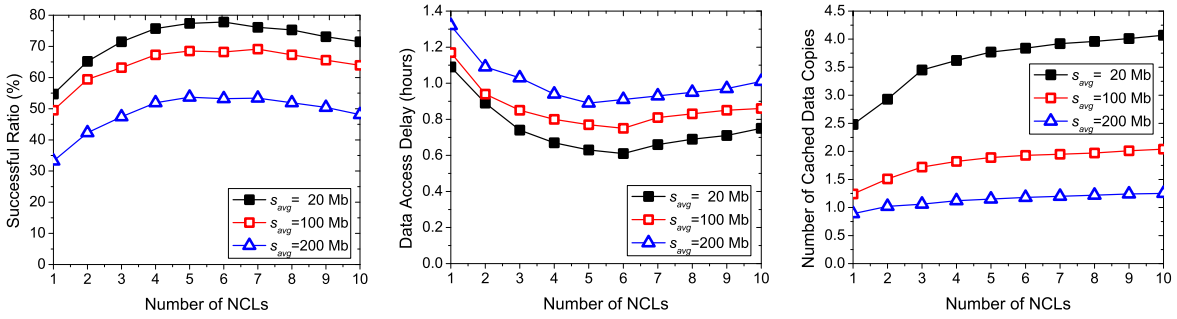
Our proposed cache replacement strategy in Section 5.4 is compared with the traditional replacement strategies including FIFO and LRU. It is also compared with Greedy-Dual-Size which is widely used in web caching.

We use *MIT Reality* trace for such evaluation, and set T as 1 week. The results are shown in Figure 15. FIFO and LRU leads to poor data access performance due to improper consideration of data popularity. In Figure 15(a), when data size is small and node buffer constraint is not tight, cache replacement will not be frequently conducted. Hence, the successful ratio of traditional strategies is only 10%-20% lower than that of our scheme. However, when data size becomes larger, these strategies do not always select the most appropriate data to cache, and the advantage of our scheme rises to over 100% when $s_{avg} = 200$Mb. Data access delay of FIFO and LRU also becomes much longer when s_{avg} increases as shown in Figure 15(b). Greedy-Dual-Size performs better than FIFO and LRU due to consideration of data popularity and size, but it is unable to ensure optimal cache replacement decision.



(a) Successful ratio (b) Data access delay (c) Cache Replacement overhead

Fig. 15. Performance of data access with different cache replacement strategies



(a) Successful ratio (b) Data access delay (c) Caching overhead

Fig. 16. Performance of data access with different number of NCLs

In Figure 15(c), we also compared the overhead of those strategies, which is the amount of data exchanged for cache replacement. Since cache replacement is only conducted locally between mobile nodes in contact, there are only slight differences of this overhead among different strategies. Greedy-Dual-Size makes the caching nodes exchange a bit more data, but this difference is generally negligible.

7.4 Number of NCLs

In this section, we investigate the impact of different numbers (K) of NCLs on data access performance using *Infocom06* trace. We set $T = 3$ hours and all the other parameters remain the same as in Section 7.2.

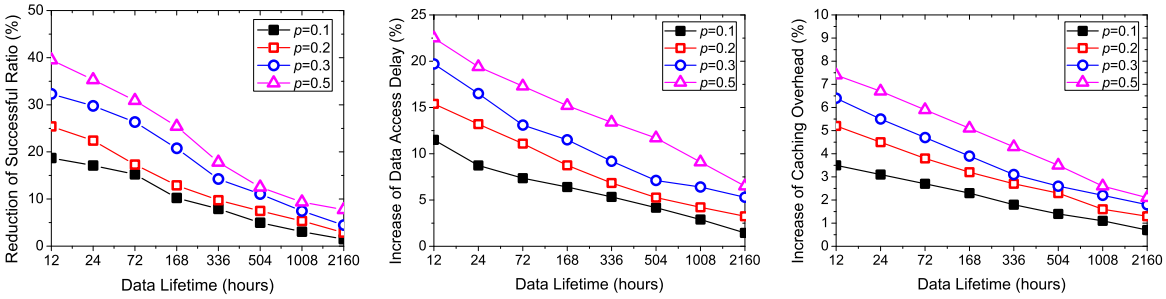
The simulation results are shown in Figure 16. When K is small, it takes longer to forward queries and data between requesters and caching nodes, and hence data access performance is reduced. This reduction is particularly significant when $K < 3$. As shown in Figures 16(a) and 16(b), when K is reduced from 2 to 1, the delivery ratio decreases by 25%, and the data access delay increases by 30%. In contrast, when K is large, further increase of K will not improve data access performance, because the newly selected central nodes are essentially not good at communicating with other nodes in the network. Meanwhile, as shown in Figure 16(c), when K is small, increasing K will consume considerably more buffer space for caching. However, this increase is negligible when K is large or node buffer constraint is tight.

In summary, when node buffer constraint is tight, smaller K is helpful to provide acceptable caching performance with lower overhead. However, too large K will not provide any extra benefit, and may even impair the performance. From Figure 16, we conclude that $K = 5$ is the best choice for *Infocom06* trace, which is consistent with the result of trace-based validation shown in Figure 4(b).

7.5 Impact of NCL Load Balancing

In this section, we evaluate the impact of NCL load balancing scheme proposed in Section 6 on caching performance. According to the evaluation results in Section 7.4, we set $K = 8$ for the *MIT Reality* trace and $s_{avg} = 100$ Mb.

Each central node periodically determines whether to migrate its functionality to another node with a fixed probability p. We set the period of making such decision to be 10% of the trace length, and the evaluation results with different values of p on the *MIT Reality* trace are shown in Figure 17. In general, when the central nodes change, the existing caching locations become inappropriate, and hence the successful ratio of data access is reduced. As shown in Figure 17(a), such reduction can be up to 40% when the data lifetime is short, but will diminish significantly to 10% when there is longer time for the queries to be forwarded to the caching nodes. Figure 17(b) also shows that the data access delay increases accordingly. Moreover, since the cached data copies are opportunistically moved to more appropriate network locations after the change of central nodes, the



(a) Reduction of successful ratio (b) Increase of data access delay (c) Increase of caching overhead

Fig. 17. Impact of NCL load balancing on the caching performance

caching overhead only slightly increases by less than 10%, as shown in Figure 17(c).

The impact of NCL load balancing is also determined by the frequency of the changes of central nodes. As shown in Figure 17, the reduction of successful ratio of data access is sensitive to the value of p. Especially when the data lifetime is short, larger value of p significantly magnify the impact on the caching performance. In general, the impact of NCL load balancing on the caching performance is largely determined by the specific network condition and data access pattern.

8 Conclusions

In this paper, we propose a novel scheme to support cooperative caching in DTNs. Our basic idea is to intentionally cache data at a set of NCLs which can be easily accessed by other nodes. We ensure appropriate NCL selection based on a probabilistic metric; our approach coordinates caching nodes to optimize the tradeoff between data accessibility and caching overhead. Extensive simulations show that our scheme greatly improves the ratio of queries satisfied and reduces data access delay, when being compared with existing schemes.

References

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN Routing As a Resource Allocation Problem. In *Proceedings of SIGCOMM*, pages 373–384, 2007.

[2] C. Boldrini, M. Conti, and A. Passarella. ContentPlace: social-aware data dissemination in opportunistic networks. In *Proceedings of MSWiM*, pages 203–210, 2008.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of INFOCOM*, volume 1, 1999.

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

[5] H. Cai and D. Y. Eun. Crossing over the bounded domain: from exponential to power-law inter-meeting time in manet. *Proc. MobiCom*, pages 159–170, 2007.

[6] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, 1997.

[7] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. *IEEE Trans. on Mobile Computing*, 6(6):606–620, 2007.

[8] P. Costa, C. Mascolo, M. Musolesi, and G. Picco. Socially Aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, 2008.

[9] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. *Proc. MobiHoc*, 2007.

[10] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. *Proc. MobiHoc*, pages 257–266, 2003.

[11] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *Proceeding of MobiSys*. ACM, 2008.

[12] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot. Diversity of Forwarding Paths in Pocket Switched Networks. In *Proceedings of IMC*, pages 161–174. ACM, 2007.

[13] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot. Delegation Forwarding. *Proc. MobiHoc*, 2008.

[14] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. *Proc. SIGCOMM*, pages 27–34, 2003.

[15] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[16] M. Fiore, F. Mininni, C. Casetti, and C. F. Chiasserini. To cache or not to cache? In *Proceedings of IEEE INFOCOM*, pages 235–243, 2009.

[17] W. Gao and G. Cao. On Exploiting Transient Contact Patterns for Data Forwarding in Delay Tolerant Networks. In *Proceedings of ICNP*, pages 193–202, 2010.

[18] W. Gao and G. Cao. User-centric data dissemination in disruption tolerant networks. In *Proceedings of INFOCOM*, 2011.

[19] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa. Supporting cooperative caching in disruption tolerant networks. In *Proceedings of Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2011.

[20] W. Gao, Q. Li, B. Zhao, and G. Cao. Multicasting in delay tolerant networks: a social network perspective. In *Proceedings of MobiHoc*, pages 299–308, 2009.

[21] Y. Huang, Y. Gao, K. Nahrstedt, and W. He. Optimizing File Retrieval in Delay-Tolerant Content Distribution Community. In *Proceedings of ICDCS*, pages 308–316, 2009.

[22] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. *Proc. MobiHoc*, 2008.

[23] S. Ioannidis, L. Massoulie, and A. Chaintreau. Distributed caching over heterogeneous mobile networks. In *Proceedings of the ACM SIGMETRICS*, pages 311–322, 2010.

[24] V. Lenders, G. Karlsson, and M. May. Wireless Ad hoc Podcasting. In *Proceedings of SECON*, pages 273–283, 2007.

[25] F. Li and J. Wu. Mops: Providing content-based service in disruption-tolerant networks. In *Proceedings of Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 526–533, 2009.

[26] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, 1990.

[27] M. J. Pitkanen and J. Ott. Redundancy and distributed caching in mobile dtns. In *Proceedings of 2nd ACM/IEEE Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*. ACM, 2007.

[28] I. Psaras, L. Wood, and R. Tafazolli. Delay-/ disruption-tolerant networking: State of the art and future challenges. *University of Surrey, Technical Report*, 2010.

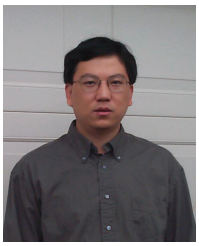
[29] J. Reich and A. Chaintreau. The Age of Impatience: Optimal Replication Schemes for Opportunistic Networks. In *Proceedings of ACM CoNEXT*, pages 85–96, 2009.

[30] S. M. Ross. *Introduction to probability models*. Academic Press, 2006.

[31] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, 2005.

[32] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Transactions on Networking*, 16(1):63–76, 2008.
[33] B. Tang, H. Gupta, and S. R. Das. Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(3):289–304, 2008.
[34] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. *Technical Report CS-200006, Duke University*, 2000.
[35] L. Yin and G. Cao. Supporting Cooperative Caching in Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 5(1):77–89, 2006.
[36] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/ subscribe communication in delay tolerant networks. *Proc. MSWiM*, pages 225–234, 2007.
[37] Q. Yuan, I. Cardei, and J. Wu. Predict and relay: an efficient routing in disruption-tolerant networks. In *Proc. MobiHoc*, pages 95–104, 2009.
[38] J. Zhao, P. Zhang, G. Cao, and C. Das. Cooperative caching in wireless p2p networks: Design, implementation, and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, pages 229–241, 2010.
[39] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni. Recognizing Exponential Inter-Contact Time in VANETs. In *Proceedings of INFOCOM*, 2010.

Wei Gao received the BE degree in electrical engineering from the University of Science and Technology of China in 2005 and the PhD degree in computer science from the Pennsylvania State University in 2012. He is currently an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. His research interests include wireless and mobile network systems, mobile social networks, cyber-physical systems, and pervasive and mobile computing. He is a member of the IEEE.



Guohong Cao received the BS degree in computer science from Xian Jiaotong University and received the PhD degree in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. He has published more than 150 papers in the areas of wireless networks, wireless security, vehicular networks, wireless sensor networks, cache management, and distributed fault tolerant computing. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS'2009, MASS'2010, and INFOCOM'2013. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.

Arun Iyengar does research and development into distributed computing, high availability, and Web performance at IBM's T.J. Watson Research Center in Yorktown Heights, NY. His techniques for caching, load balancing, and serving dynamic content are widely used for Web and distributed applications. He is Founding Co-Editor-in-Chief of the ACM Transactions on the Web, Chair of IFIP Working Group 6.4 on Internet Applications Engineering, and an IBM Master Inventor. He has a PhD in computer science from the Massachusetts Institute of Technology (MIT). He is a Fellow of the IEEE.

Mudhakar Srivatsa is a Research Staff Member in Network Technologies Department at Thomas J. Watson Research Center. He received his PhD in Computer Science from Georgia Tech. His research interests primarily include network analytics and secure information flow. He serves as a technical area leader for Secure Hybrid Networks research in International Technology Alliance in Network and Information Sciences and as a principal investigator for Information Network Research in Network Science Collaborative Technology Alliance. He is a member of the IEEE.