2011 31st International Conference on Distributed Computing Systems

Supporting Cooperative Caching in Disruption Tolerant Networks

Wei Gao and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
{*weigao,gcao*}*@cse.psu.edu*

Arun Iyengar and Mudhakar Srivatsa
IBM T. J. Watson Research Center
Hawthorne, NY 10532
{*aruni, msrivats*}*@us.ibm.com*

***Abstract*—Disruption Tolerant Networks (DTNs) are characterized by the 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 effective 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 effective scheme which ensures appropriate NCL selection based on a probabilistic selection metric, and coordinate multiple caching nodes to optimize tradeoff between data accessibility and caching overhead. Extensive trace-driven simulations show that our scheme significantly improves data access performance compared to existing schemes.**

I. Introduction

Disruption Tolerant Networks (DTNs) [7] consist of mobile devices which contact each other opportunistically. Due to the low node density and unpredictable node mobility, only intermittent connectivity among mobile nodes exist in DTNs, and the subsequent difficulty of maintaining persistent end-to-end connection makes it necessary to use "carry-and-forward" methods for data transmission. Node mobility is exploited to let mobile nodes physically carry data as relays, and forward data opportunistically when contacting others. The key problem is therefore how to determine the appropriate relay selection strategy.

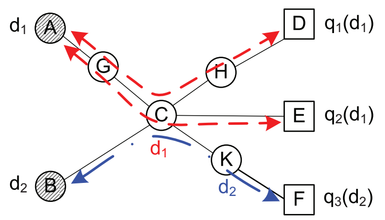
Although many data forwarding schemes have been proposed in DTNs [4], [1], [6], [12], [10], there is only limited research effort on providing effective data access to mobile users, despite the importance of data accessibility in many mobile computing applications. For example, it is desirable that Smartphone 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 will only be requested by mobile users whenever needed, and requesters do not know the data locations in advance. The destination of data is hence unknown when data is generated. This communication paradigm differs from well-studied publish/subscribe systems [24], [16], 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 the requesters in such cases.

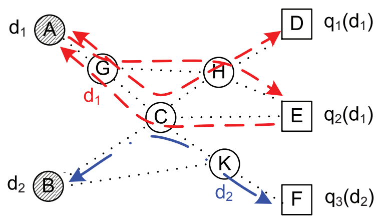
A common technique used to improve data access performance is caching, i.e., to cache data at appropriate network locations based on the query history, so that queries in the future can be responded with less delay. Although cooperative caching has been extensively studied for both web-based applications [8], [22] and wireless ad-hoc networks [23] to allow the sharing and coordination of cached data among multiple 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 about data transmission delay, and furthermore makes it difficult to determine appropriate caching location 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 effectively 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 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 effective approach to NCL selection in DTNs based on a probabilistic selection metric.
- We propose a data access scheme to probabilistically coordinate multiple caching nodes for responding to user queries, and furthermore optimize the tradeoff between data accessibility and caching overhead.
- 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.



(a) Wireless ad-hoc networks



(b) Disruption-Tolerant Networks

Figure 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.

The rest of this paper is organized as follows. In Section II we briefly review existing work. Section III provides an overview of our approach and highlights our motivation of intentional caching in DTNs. Section IV describes how to appropriately select NCLs in DTNs, and Section V describes the details of our proposed caching scheme. The results of trace-driven performance evaluations are shown in Section VI, and Section VII concludes the paper.

II. Related Work

Research on data forwarding in DTNs originates from Epidemic routing [21] which floods the entire network. Some later studies develop relay selection strategies to approach the performance of Epidemic routing with lower forwarding cost, based on prediction of node contact in the future. Some schemes do such prediction based on their mobility patterns, which are characterized by semi-Markov chains [25] or Hidden Markov Models [9]. Some others [4], [1] exploit node contact records in the past as stochastic processes for better prediction accuracy.

Data access in DTNs, on the other hand, can be provided in various ways. Data can be disseminated to appropriate users based on their interest profiles [11]. Publish/subscribe systems [24], [16] are most commonly used for such data dissemination, and they usually exploit social community structures to determine the brokers. In other schemes [15], [2] without brokers, data items are grouped into pre-defined channels, and data dissemination is based on the users' subscriptions to these channels.

Caching is another way to provide data access. In [23], the authors studied cooperative caching in wireless ad-hoc networks, in which each node caches pass-by data based on data popularity, so that queries in the future can be responded with less delay. They selected caching locations incidentally among all the network nodes. Some research efforts [18], [13] have been made for caching in DTNs, but they only improve data accessibility from infrastructure network such as WiFi Access Points (APs) [13] or Internet [18]. 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 [19], [14], but they generally rely on assumptions which simplify the network conditions. In [19], it is assumed that all the nodes in the network contact each other with the same rate. In [14], mobile users are artificially partitioned into several classes, such that users in the same class are statistically identical. Comparatively, in this paper we propose to support cooperative caching among peer mobile nodes in DTNs with heterogeneous contact patterns and behaviors.

III. Overview

A. Motivation

A requester generally queries the network to access a data item. The data source or caching nodes then 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 the key constraint is 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 existence of 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. Therefore, any intermediate node on the path is able to cache the pass-by data. For example, in Figure 1(a), C forwards all the three queries to the data sources A and B, and also forwards the 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 the query history, and similarly data d_2 is cached at node K. In general, any node in the network is able to cache the pass-by data incidentally.

However, the effectiveness of such incidental caching strategy is seriously impaired in DTNs due to the lack of persistent network connectivity. Since data is forwarded via opportunistic contacts, the query and replied data may take different routes, which makes it difficult for nodes to collect the information about query history and make caching decision. For example, in Figure 1(b), after having

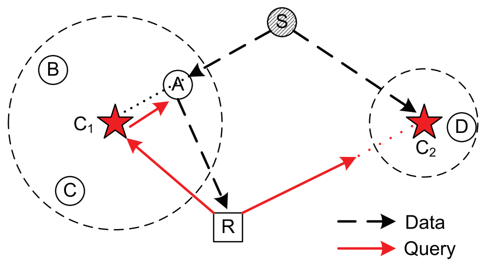


Figure 2. The big picture of intentional caching

forwarded query q_2 to the data source A, node C may lose its connection to G, and hence does not have chance to 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 the query q_2 and considers d_1 less popular. In this case, d_1 will eventually be cached at node G, and hence needs longer time to be replied to the requester.

Our basic solution to address this challenge and improve the 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 the constrained scope of caching locations then reduces the complexity of maintaining query history and making appropriate caching decision.

B. Network Model

Opportunistic contacts in DTNs are described by network *contact graph* $G(V, E)$, where stochastic contact process between a node pair $i, j \in V$ is modeled as an edge e_{ij}. The characteristics of an edge $e_{ij} \in E$ are mainly determined by the properties of inter-contact time among mobile nodes. Similar to previous work [1], [26], we consider the pairwise node inter-contact time as exponentially distributed. The contacts between nodes i and j then form a Poisson process with contact rate λ_{ij}, which remains relatively constant and is calculated at real-time from the cumulative contacts between nodes i and j since the network starts.

C. The Big Picture

We consider a general caching scenario, in which each node may generate data with a globally unique identifier and finite lifetime, and may also request for another data by sending queries with a finite time constraint. Therefore, data requesters are randomly distributed in the network and are not spatially correlated with each other. We focus on effectively utilizing the available node buffer to optimize the overall caching performance indicated by data access delay.

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 NCLs for data access[1]. The big picture of our proposed scheme is illustrated in Figure 2. Each NCL is represented by a central node[2], 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, which is illustrated as the dashed circles in Figure 2. 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 the data. In this case, data accessibility is determined by both node contact frequency and data lifetime.

IV. 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. The applicability of such selection in practice is supported by the heterogeneity of node contact pattern in realistic DTN traces.

A. 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.

The inter-contact time X_k between nodes N_k and N_{k+1} on P_{AB}, as a random variable, follows an exponential distribution with probability density function (PDF) $p_{X_k}(x) = \lambda_k e^{-\lambda_k x}$. Hence, the total time needed to transmit data from A to B is $Y = \sum_{k=1}^{r} X_k$ following a hypoexponential distribution [20], such that

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

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

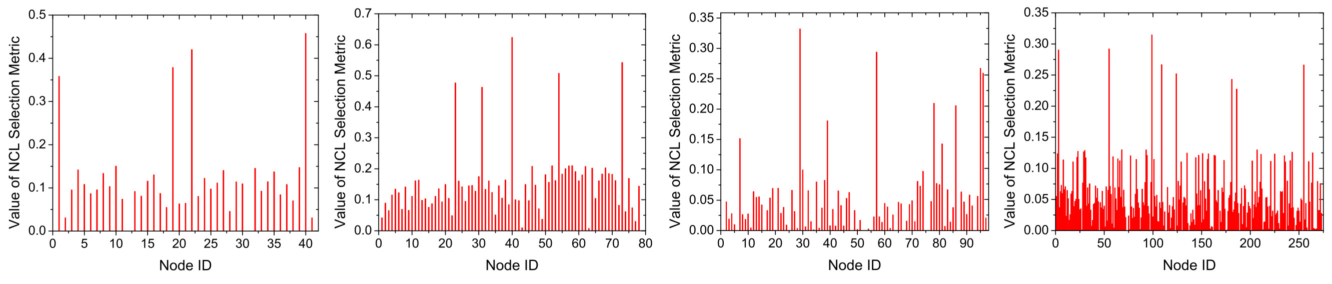
$$p_{AB}(T) = \int_0^T p_Y(x)dx = \sum_{k=1}^{r} C_k^{(r)} \cdot (1 - e^{-\lambda_k T}), \quad (2)$$

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

[2]In the rest of this paper, a central node is used equivalently to denote the corresponding NCL.

Table I
TRACE SUMMARY

Trace	Infocom05	Infocom06	MIT Reality	UCSD
Network type	Bluetooth	Bluetooth	Bluetooth	WiFi
No. of devices	41	78	97	275
No. of internal contacts	22,459	182,951	114,046	123,225
Duration (days)	3	4	246	77
Granularity (secs)	120	120	300	20
Pairwise contact frequency (per day)	4.6	6.7	0.024	0.036



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

Figure 3. Values of NCL selection metric on realistic DTN traces

and the data transmission delay between two nodes A and B 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 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:

$$C_i = \frac{1}{N - |\mathbb{N}_C|} \cdot \sum_{j \in V \setminus \mathbb{N}_C} p_{ij}(T), \qquad (3)$$

where $N = |V|$, $p_{ii}(T) = 0$ and $\mathbb{N}_C$ indicates the set of selected central nodes. This metric indicates the average probability that data can be transmitted from a random non-central node to node i within time T.

Central nodes representing NCLs are selected sequentially by the network administrator before data access, based on the global knowledge about pairwise contact rates and shortest opportunistic paths among mobile nodes. Each time the node in $V \setminus \mathbb{N}_C$ with the highest metric value is selected as the central node, until the required K central nodes are selected. K is a pre-defined parameter determined by performance requirements and will be discussed in Section V-E in more details. Note that in Eq. (3) the existing central nodes are excluded from calculating C_i, the selected central nodes hence will not be clustered on network contact graph.

A network warm-up period is reserved for nodes to collect information and calculate their pairwise contact rate as described in Section III-B. The parameter T used in Eq. (3) is determined by the average node contact frequency in the network. This parameter is generally trace-dependent and will be discussed later in Section IV-B.

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, and this notification can be done via cellular 3G links. Since each node is only notified about the list of central nodes, this notification is cost-effective without producing noticeable communication overhead.

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. According to Section III-B, the pairwise contact rates among nodes tends to remain stable during long time, and hence the selected NCLs do not need to be changed during data access. Since central nodes are selected in a sequential manner, a selected central node unwilling to cache data can be simply neglected without affecting the selection of other central nodes.

B. Trace-based Validation

The practical applicability of NCL selection 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 handheld mobile devices in corporate environments, including conference sites and university campus. The devices equipped with a Bluetooth interface periodically detect their peers nearby, and a contact is recorded when two devices move close to each other. The devices equipped with a WiFi interface search for nearby WiFi Access Points (APs) and associate themselves to the APs with the best signal strength. A contact is recorded when two devices are associated to the same AP. The traces are summarized in Table I.

We calculate pairwise node contact rates based on their cumulative contacts during the entire trace. According to Eq. (2), inappropriate values of T will make C_i close to 0 or 1. Instead, the values of T are adaptively determined in different traces to ensure the differentiation of NCL selection metric values of mobile 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 3 show that the distributions of NCL selection metric values of nodes are highly skewed in all traces, such that the metric values of few nodes are much higher than that of others. This difference can be up to tenfold, and supports that our proposed NCL selection metric appropriately indicates the heterogeneity of node contact pattern. As a result, the selected NCLs can be easily accessed by other nodes in the network.

V. 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 be decided to 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 the tradeoff between data accessibility and transmission overhead by controlling the number of returned data copies.
3) Utility-based[3] cache replacement is conducted whenever two caching nodes contact each other, 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, and probabilistically cache less popular data to ensure the overall data accessibility.

A. 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 higher metric than itself.

[3]For newly created data, the utility value will initially be low since the data has not yet been requested.

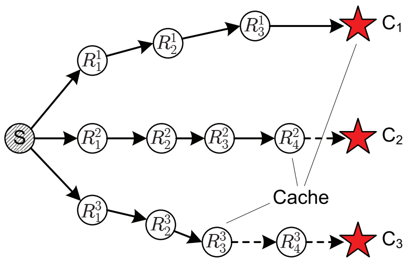


Figure 4. Determining caching location at NCLs

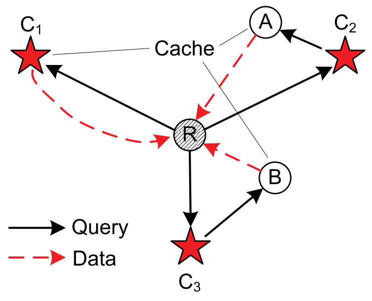


Figure 5. Pulling data from the NCLs

For newly generated data, the initial caching locations are automatically determined during such forwarding process based on node buffer conditions. The caching locations are then dynamically adjusted by cache replacement described in Section V-D 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 4, 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.

B. Queries

We assume that any node may request data, and hence data requesters are randomly distributed in the network. A requester multicasts query with a finite time constraint to all the central nodes to pull data, and existing multicast schemes in DTNs [12] can be exploited for this purpose.

After having received the query, a central node immediately replies to the requester with the data if it is cached

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

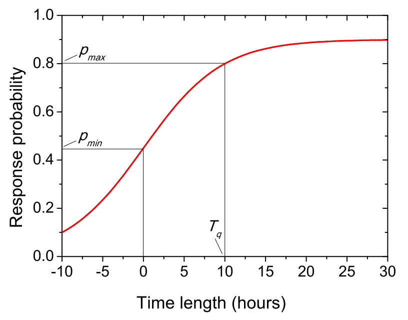


Figure 6. Probability for deciding data response

locally[5]. Otherwise, it broadcasts query to the nodes nearby. This process is illustrated in Figure 5. 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 have received the query from central nodes C_2 and C_3, respectively. The query broadcast finishes when the query expires, so that each caching node at the NCLs is able to maintain the up-to-date information about query history, which is used in Section V-D for cache replacement.

C. Probabilistic Response

As shown in Figure 5, multiple data copies are replied to the requester from NCLs to ensure that the requester receives data before the 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 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.

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 the query is generated with a time constraint T_q, and it takes $t_0 < T_q$ for the query 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 the 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 $T_q - t_0$. In general, 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

$$p_R(t) = \frac{k_1}{1 + e^{-k_2 \cdot t}}, \tag{4}$$

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. (4) 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 6.

D. Cache Replacement

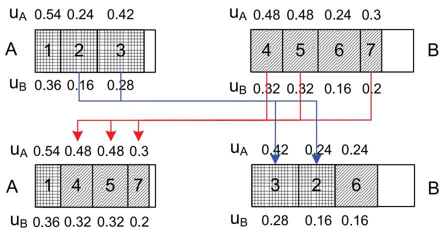
Caching locations of data are dynamically adjusted via cache replacement. This replacement is based on data popularity, and places popular data nearer to the central nodes of NCLs. Traditional cache replacement strategies such as LRU, which removes the least-recently-used data from cache when new data is available, are ineffective due to its oversimplistic consideration of data popularity. Greedy-Dual-Size [5] 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, and heuristically balances between data accessibility and access delay.

1) ***Data Popularity:*** The popularity of a data item is probabilistically estimated based on the past k requests to this data happened 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 $w_i = 1 - e^{-\lambda_d \cdot (t_e - t_k)}$.

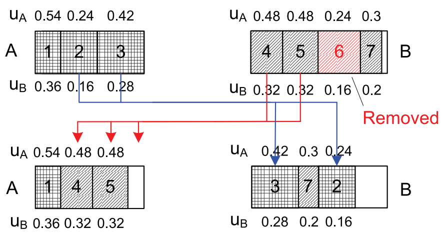
2) ***Basic Strategy:*** Cache replacement opportunistically occurs whenever two caching nodes A and B contact each other, and they exchange their cached data to optimize the cumulative data access delay[6]. We collect the cached data at both nodes into a selection pool $\mathbb{S} = \{d_1, ..., d_n\}$, and formulate cache replacement as the following knapsack problem:

[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 center node.

[6]In DTNs, since nodes are only able to 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

Figure 7. Cache replacement

$$
\begin{aligned}
&\max \sum_{i=1}^{n} x_i u_i + \sum_{j=1}^{n} y_j v_j \\
&\text{s.t.} \sum_{i=1}^{n} x_i s_i \le S_A, \sum_{j=1}^{n} y_j s_j \le S_B \\
&\quad x_i + y_i \le 1, \text{ for } \forall i \in [1, n],
\end{aligned}
\tag{5}
$$

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 is the size of caching buffer of node 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 node 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 of A and B.

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 $p_A > p_B$. In such cases, node A is prioritized to select its data to cache from the selection pool $\mathbb{S}$, by solving the following knapsack problem extracted from Eq. (5):

$$
\begin{aligned}
&\max \sum_{i=1}^{n} x_i u_i \\
&\text{s.t.} \sum_{i=1}^{n} x_i s_i \le S_A.
\end{aligned}
\tag{6}
$$

Afterwards, node B selects data to cache from the remaining part of $\mathbb{S}$ by solving a similar problem to Eq. (6). Since S_A and s_i in Eq. (6) 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 [17].

This replacement process is illustrated by an example shown in Figure 7, 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 unused caching buffer of both nodes is left blank in the figure. 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 7(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 7(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.

*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 III-B, 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 7(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.

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 the 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 knapsack problem in Eq. (6) 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.

Algorithm 1: Probabilistically 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
            $x_{i_{\max}} = 1$
            $\mathbb{S} = \mathbb{S} \setminus d_{i_{\max}}$
            $S_A = S_A - s_{i_{\max}}$, $V_{\max} = V_{\max} - s_{i_{\max}}$
        $\mathbb{S}' = \mathbb{S}' \setminus d_{i_{\max}}$
    $i_{\min} = \arg\min_i \{s_i | d_i \in \mathbb{S}, x_i == 0\}$
```

E. 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. The major reason for such delay is opportunistic network connectivity in DTNs, which can be a result of node mobility, device power outage or malicious attacks. In other words, the effects of node mobility to caching performance is implicitly considered by exploiting node contact process.

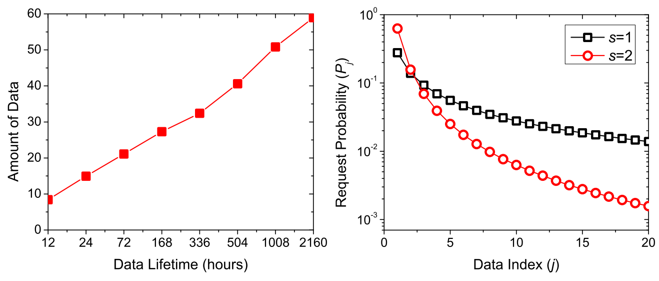
Data access delay is closely related to the number (K) of NCLs. When K is small, the average distance from a node to the NCLs is longer, which makes the first and third parts of the delay bigger. Meanwhile, since the total amount of data being cached in the network is small, data is more likely to be 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 effective. 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 VI-D, we will experimentally investigate the impact of different values of K to the caching performance in more details.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed intentional caching scheme, which is compared with the following data access schemes:



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

Figure 8. Experiment setup

- **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** [23], which is proposed for cooperative caching in wireless ad-hoc networks, and lets each selected relay in DTNs cache the pass-by data according to their popularity.
- **Bundle Cache** [18], 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.

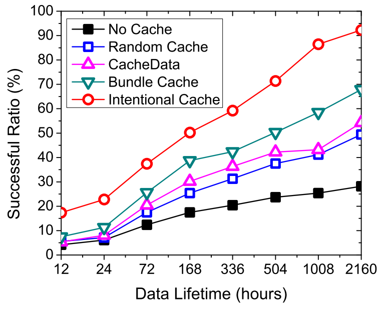
Cache replacement algorithms are proposed in CacheData and Bundle Cache, and will also be used in our evaluations. For No Cache and Random Cache, LRU is used for cache replacement. The following metrics are used for evaluations. Each simulation is repeated multiple times with randomly generated data and queries for statistical convergence.

- **Successful ratio**, the ratio of queries being satisfied with the requested data.
- **Data access delay**, the average delay for getting responses to queries.
- **Caching overhead**, the average number of data copies being cached in the network.

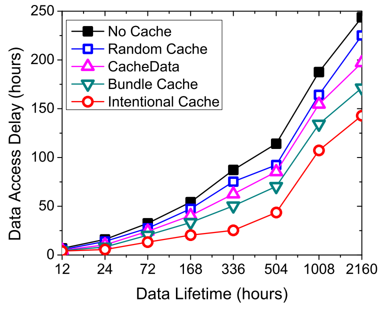
A. Experiment Setup

Our performance evaluations are performed on the *Infocom06* and *MIT Reality* traces. In all the experiments, 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.

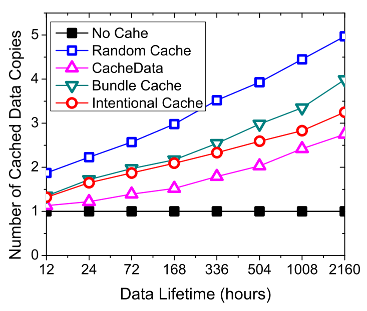
*1) **Data Generation**:* Each node periodically checks whether it has generated data which has not expired yet. If not, it determines whether to generate new data with probability p_G. Each generated data has finite lifetime uniformly distributed in range $[0.5T, 1.5T]$, and the period for data generation 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



(a) Successful ratio

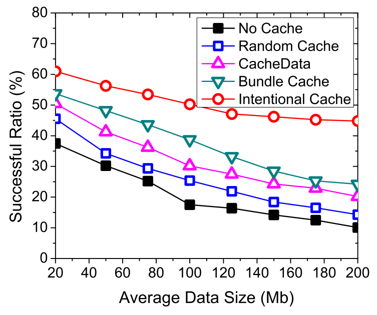


(b) Data access delay

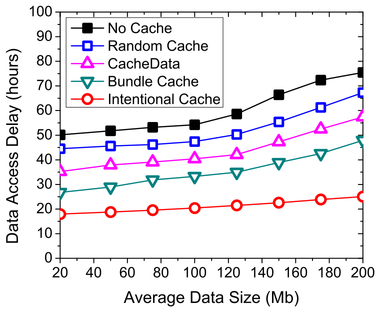


(c) Caching overhead

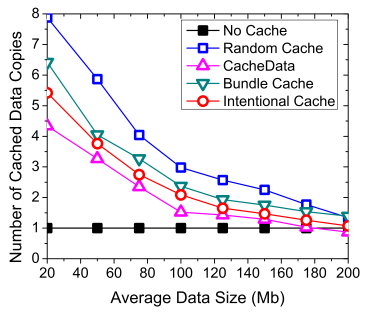
Figure 9. Performance of data access with different data lifetime



(a) Successful ratio



(b) Data access delay



(c) Caching overhead

Figure 10. Performance of data access with different node buffer conditions

controlled by T, as illustrated in Figure 8(a) for the *MIT Reality* trace. Similarly, data size is uniformly distributed in range $[0.5s_{avg}, 1.5s_{avg}]$, and caching buffer of nodes is uniformly distributed in range [200Mb, 600Mb]. s_{avg} is adjusted to simulate different node buffer conditions.

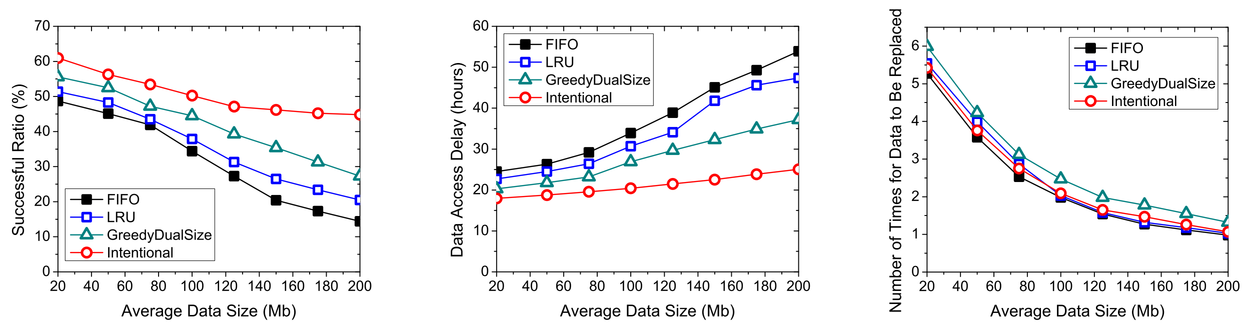
*2) **Query Pattern**:* Queries are randomly generated at all nodes, and each query has a finite time constraint $T/2$. We assume that query pattern follows Zipf distribution which has been proved to describe the query pattern of web data access [3]. Let $P_j \in [0, 1]$ be the probability that data j is requested, and 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 8(b). Every time $T/2$, each node determines whether to request data j with probability P_j.

B. Caching Performance

We first evaluate the caching performance of our scheme using the *MIT Reality* trace. We set the number (K) of NCLs as 8 and generate query pattern following 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.

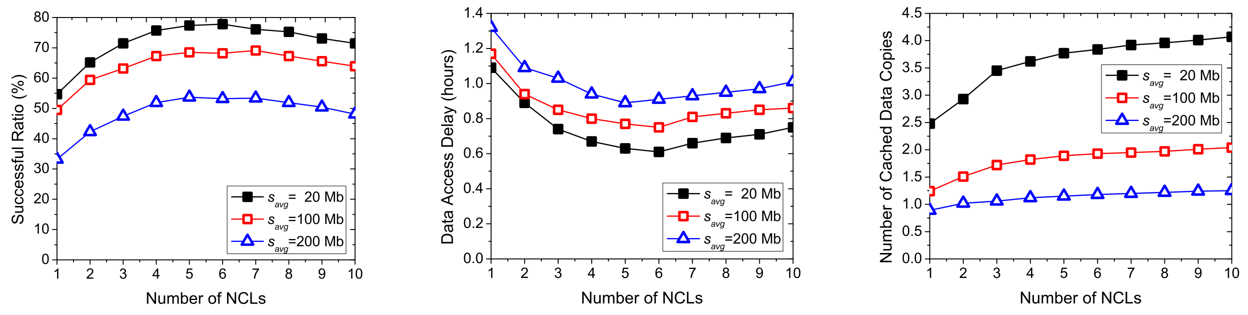
The simulation results with different values of T are shown in Figure 9. The successful ratio of data access is mainly restrained by T itself. When T increases from 12 hours to 3 months, the successful ratio of all schemes is significantly improved, because data has more time to be delivered to requesters before expiration. Since the selected NCLs are effective in communicating with other nodes, our proposed intentional caching scheme achieves much better successful ratio and delay of data access. As shown in Figures 9(a) and 9(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 to be used in DTNs due to the difficulty of maintaining query history.

Meanwhile, Figure 9(c) shows that our proposed 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 seriously 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



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

Figure 11. Performance of data access with different cache replacement strategies



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

Figure 12. Performance of data access with different number of NCLs

cost is proportional to data access delay during which data is carried by relays. Hence, the cost-effectiveness of our scheme is strongly supported by Figure 9(b).

We also evaluated data access performance with different node buffer conditions, which is realized by adjusting s_{avg}. The simulation results are shown in Figure 10. When data size becomes larger, less data can be cached as shown in Figure 10(c), and data access performance is hence reduced. In Figures 10(a) and 10(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 in the limited cache space.

C. *Effectiveness of Cache Replacement*

In this section, we evaluate the effectiveness of our proposed cache replacement strategy in Section V-D for improving data access performance. Our proposed strategy 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 simulation results are shown in Figure 11. FIFO and LRU leads to poor data access performance due to improper consideration of data popularity. In Figure 11(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 11(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.

In Figure 11(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.

D. *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 VI-B.

The simulation results are shown in Figure 12. 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 12(a) and 12(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 12(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, using smaller value of K is helpful to provide acceptable caching performance with lower overhead. However, selecting too many NCLs will not provide any extra benefit, and may even impair the caching performance. From Figure 12, we generally conclude that $K = 5$ is the best choice for the *Infocom06* trace, which is surprisingly consistent with the result of trace-based validation shown in Figure 3(b).

VII. 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 pre-specified set of NCLs which can be easily accessed by other nodes. We propose an effective scheme which ensures appropriate NCL selection based on a probabilistic selection metric, and furthermore coordinates multiple caching nodes to optimize the tradeoff between data accessibility and caching overhead. Extensive trace-driven simulations show that our scheme significantly improves the ratio of queries satisfied and reduces data access delay, compared with existing caching schemes.

Acknowledgment

We would like to thank Jie Liu and the anonymous reviewers for their insightful comments and helpful suggestions. This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-0721479, and by Network Science CTA under grant W911NF-09-2-0053.

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

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

[7] K. Fall. A delay-tolerant network architecture for challenged internets. *Proc. SIGCOMM*, pages 27–34, 2003.

[8] 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.

[9] W. Gao and G. Cao. Fine-Grained Mobility Characterization: steady and transient state behaviors. In *Proceedings of MobiHoc*, pages 61–70, 2010.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] V. Lenders, G. Karlsson, and M. May. Wireless ad hoc podcasting. In *Proceedings of SECON*, pages 273–283, 2007.

[16] F. Li and J. Wu. Mops: Providing content-based service in disruption-tolerant networks. In *Proceedings of ICDCS*, pages 526–533, 2009.

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

[18] 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.

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

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

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

[22] D. Wessels and K. Claffy. ICP and the Squid Web Cache. *IEEE Journal on Selected Areas in Communications (JSAC)*, 16(3):345–357, 2002.

[23] L. Yin and G. Cao. Supporting Cooperative Caching in Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 5(1):77–89, 2006.

[24] 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.

[25] Q. Yuan, I. Cardei, and J. Wu. Predict and relay: an efficient routing in disruption-tolerant networks. In *Proc. MobiHoc*, pages 95–104, 2009.

[26] 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.