Supporting Cooperative Caching in Disruption Tolerant Networks

Wei Gao*, Guohong Cao†, Arun Iyengar‡ and Mudhakar Srivatsa†
*Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA
†IBM T. J. Watson Research Center, Hawthorne, NY
‡{weigao, gcao}@cse.psu.edu, †{aruni, mshrivats}@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 the 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. To address the challenges of opportunistic network connectivity in DTNs, 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 furthermore coordinate multiple caching nodes to optimize the tradeoff between data accessibility and caching overhead. By extensive trace-driven simulations, we show that our proposed caching scheme significantly improves the performance of data access, in aspects of the successful ratio of queries and data access delay, compared to existing schemes.

I. INTRODUCTION

Disruption Tolerant Networks (DTNs) [11] 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. More specifically, node mobility is exploited to let mobile nodes physically carry data as relays, and forward data opportunistically upon contacts with others. The key problem is therefore how to determine the appropriate relay selection strategy.

Although a large variety of data forwarding schemes have been proposed in DTNs [26], [4], [1], [10], [13], there is only limited research effort on providing effective data access to mobile users [4] in such challenging networks, despite the importance of data accessibility in many mobile computing applications. For example, with the popularization of Smartphones, it is desirable that mobile users can find interesting digital content from their nearby peers. In Vehicular Ad-hoc Networks (VANETs), the availability of live traffic information about specific road segments will be beneficial for nearby vehicles to avoid traffic delays.

In these applications, data will only be requested by mobile users whenever needed, and the data requesters do not know the locations of data in the network in advance. As a result, the destination of data is unknown when the data is generated.

†In the rest of this paper, “node” and “user” are used interchangeably.

This communication paradigm significantly differs from that of the well-studied publish/subscribe systems [30], [1], [21], in which data is automatically forwarded by broker nodes to users according to their data subscriptions. Appropriate network design is therefore needed to ensure that data can be promptly accessed by the requesters in such cases. A common technique used to improve the performance of data access is caching. The basic idea is 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 [12], [28] and wireless ad-hoc networks [29] 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 among mobile nodes, and furthermore makes it difficult to determine the appropriate caching location for reducing the data access delay. This difficulty is also raised by the incomplete information at individual mobile nodes about the query history. Second, since data can only be transmitted via opportunistic contacts, multiple data copies need to be cached at different nodes to ensure data accessibility. The difficulty in coordinating multiple caching nodes makes it hard to achieve the best 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. More specifically, 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 in caching, and we ensure that popular data is always cached nearer to the central nodes via dynamic cache replacement based on the query history. Our detailed contributions are listed as follows:

- We develop an effective approach to appropriate 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 an utility-based cache replacement scheme.
to dynamically adjust the locations of cached data based on the query history, and our scheme achieves good tradeoff between the cumulative data accessibility and access delay.

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 [27] 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 the prediction of node contact in the future. Some schemes do such prediction by estimating node co-location probabilities based on their mobility patterns, which are characterized by Kalman filter [8] or semi-Markov chains [31]. Some others [4], [1] exploit node contact records in the past as stochastic process for better prediction accuracy, based on the experimental [2], [18] and theoretical [5] analysis on node contact characteristics. The social network properties of human mobility, such as the centrality and community structures, are also exploited for data forwarding decision in recent social-based forwarding schemes [9], [15], [13].

Data accessibility to mobile users in DTNs, on the other hand, can be provided in various ways. For example, after having been generated, data can be disseminated to the appropriate users based on their interest profiles maintained in the network. Publish/subscribe systems [30], [21] are most commonly used for such data dissemination. In these schemes, social community structures are usually exploited to determine the brokers, and the effectiveness of data access is then improved via differentiation of data access strategies across the community boundary. In other schemes [20], [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 effective data access. [29] studied cooperative caching in wireless ad-hoc networks, in which each node caches pass-by data based on the data popularity, so that queries in the future can be responded by the caching node with less delay. In other words, the caching locations are selected incidentally among all the nodes in the network. Some research efforts [23], [14] have also been made for caching in DTNs, but the proposed caching strategies only improve data accessibility from the infrastructure network, such as WiFi Access Points (APs) [14] or Internet gateways [23]. The peer-to-peer data sharing and access among mobile users themselves are generally neglected.

Distributed determination of appropriate caching policies for minimizing the data access delay has also been studied in DTNs [24], [17], but they generally rely on specific assumptions to simplify the network conditions. For example, in [24], it is assumed that all the mobile nodes in the network contact each other with the same rate. [17] artificially partitions mobile users into several classes, such that users in the same class are statistically identical.

III. OVERVIEW

A. Motivation

A requester generally queries the network to access a specific data item. The data source then replies to the requester with the data, after having received the query. Based on this communication paradigm, the key difference between caching strategies in wireless ad-hoc networks and DTNs is illustrated in Figure 1 where data $d_1$ generated by node $A$ is requested by nodes $D$ and $E$, and data $d_2$ generated by node $B$ is requested by node $F$. A solid line in Figure 1(a) indicates a wireless link, and a dotted line in Figure 1(b) indicates that the two nodes opportunistically contact each other. Note that the basic prerequisite is that each node has only limited buffer for caching. Otherwise, data can be cached everywhere in the network, 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. Although such paths may change over time due to node mobility, the network is assumed to be connected at any time, 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 buffer, $C$ 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 through opportunistic node contacts, the query and replied data may take different routes, which makes it difficult for the mobile nodes to collect the information about query history and furthermore make caching decision. For example, in Figure 1(b), after having forwarded query $q_2$ to the data source $A$, node $C$ may lose its connection to $G$, and therefore does not have the chance to cache data $d_1$ replied to requester $E$. On the other hand, 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 its accessibility to the requesters is reduced.

Our basic solution to address the aforementioned problems and improve the caching performance in DTNs is to restrain the scope of nodes being involved into 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 Modeling

Opportunistic node contacts in DTNs are described by the network contact graph $G(V,E)$, where the stochastic contact process between a node pair $i,j$ in $V$ is modeled as an edge $e_{ij}$. 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 mainly determined by the properties of inter-contact time among mobile nodes. Similar to previous work [11], [19], [16], [33], we consider the pairwise node inter-contact time as exponentially distributed. The contacts between nodes $i$ and $j$ then form a Poisson process with the contact rate $\lambda_{ij}$, which remains relatively constant and is calculated at real-time from the cumulative contacts between nodes $i$ and $j$ in a time-average manner.

C. The Big Picture

In this paper, we consider a general application scenario for cooperative caching in DTNs, in which each node may generate data with a globally unique identifier and specific 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 assume that each node has only limited buffer for caching, and our objective is to effectively utilize the available buffer to optimize the overall caching performance, which is indicated by the 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. Correspondingly, queries are forwarded to these NCLs for data access. Note that our scheme is different from publish/subscribe system, in which the published data is forwarded immediately by the brokers to the subscribers and will not be cached in the meantime.

The big picture of our proposed caching scheme is illustrated in Figure 2. Each NCL is represented by a central node, which corresponds to a red 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 $C_2$ at the 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 $C_1$. In general, multiple nodes at a NCL may be involved for caching, and a NCL corresponds to a connected subgraph of the network contact graph $G$, which is illustrated as the dashed circles in Figure 2. Correspondingly, the requester $R$ pulls the data by querying the NCLs, and data copies from multiple NCLs are returned to the requester, in order to ensure data accessibility within the time constraint of the query. Particularly, some NCL such as $C_2$ in Figure 2 may be too far from $R$ to receive the query on time, and does not respond with the data.

IV. NETWORK CENTRAL LOCATIONS

In this section, we describe how to appropriately select NCLs based on a probabilistic metric evaluating the data transmission delay among mobile nodes in DTNs. The applicability of such selection in practice is then validated by the heterogeneity of node contact pattern in realistic DTN traces.

A. NCL Selection Metric

In order to develop an appropriate metric for NCL selection, we first define the multi-hop opportunistic connection on the 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 $\{\lambda_1, \lambda_2, ..., \lambda_r\}$. The

\[^2\text{In the rest of this paper, a central node is used equivalently to denote the corresponding NCL.}\]
path weight is the probability \( p_{AB}(T) \) that a data item is opportunistically transmitted from A to B along \( P_{AB} \) within time \( T \).

An opportunistic path is illustrated in Figure 3. The inter-contact time \( X_k \) between nodes \( N_k \) and \( N_{k+1} \), 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 \( T = \sum_{k=1}^{r} X_k \) following a hypoexponential distribution \([23]\), such that

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

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

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

\[
p_{AB}(T) = \int_{0}^{T} p_Y(x) dx = \sum_{k=1}^{r} C_k^{(r)} \cdot (1 - e^{-\lambda_k T}),
\]

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.

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

\[
C_i = \frac{1}{N-1} \cdot \sum_{j=1, j \neq i}^{N} p_{ij}(T),
\]

where \( N \) is the total number of nodes in the network. This metric indicates the average probability that data can be transmitted from a random node in the network to node \( i \) within time \( T \), and therefore can also be considered as indicating the average distance from a random node in the network to node \( i \).

In practice, the top \( K \) nodes with the highest metric values are selected by the network administrator as the central nodes of NCLs, and such NCL selection is done before any data access operation. The number \( (K) \) of NCLs is a pre-defined network parameter. A network warm-up period is needed for the administrator to collect information about the pairwise node contact rate, and to calculate the weight of opportunistic path among mobile nodes. The NCL information is notified by the administrator to each node in the network, and a node maintains its shortest opportunistic path to each NCL.

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 the network modeling in Section II-B, node contact pattern represents the long-term mobility characteristics of nodes in DTNs, such that the pairwise contact rate tends to remain stable during a long period of network execution time. This stability is also validated in various mobility scenarios \([13], [33]\). As a result, the selected NCLs will not be changed during data access.

B. Trace-based Validation

The practical applicability of NCL selection is based on the heterogeneity of node contact patterns. More specifically, mobile nodes in DTNs differ in their popularity, such that few nodes contact many others and act as the communication hubs in the network. 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 Bluetooth interface periodically detect their peers nearby, and a contact is recorded when two devices move close to each other. The devices equipped with 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.

In order to calculate the weight of an opportunistic path according to Eq. (2), we calculate the pairwise contact rates based on the cumulative contacts between each pair of nodes during the entire trace. According to Eq. (3), inappropriate values of \( T \) will make \( C_i \) close to 0 or 1. Therefore, due to the heterogeneity of the pairwise contact frequency in different traces, different values of \( T \) are used adaptively in such calculation to ensure the differentiation of the 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 4 show that the distributions of NCL selection metric values of mobile nodes are highly skewed in all traces, such that the metric values of a few nodes are much higher than that of other nodes. This difference can be up to tenfold in some traces, and validates that our proposed NCL selection metric appropriately reflects the heterogeneity of node contact pattern. As a result, the selected NCLs can be

<table>
<thead>
<tr>
<th>Trace</th>
<th>Infocom05</th>
<th>Infocom06</th>
<th>MIT Reality</th>
<th>UCSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network type</td>
<td>Bluetooth</td>
<td>Bluetooth</td>
<td>Bluetooth</td>
<td>WiFi</td>
</tr>
<tr>
<td>No. of devices</td>
<td>41</td>
<td>78</td>
<td>97</td>
<td>275</td>
</tr>
<tr>
<td>No. of internal contacts</td>
<td>22,459</td>
<td>182,951</td>
<td>114,046</td>
<td>123,225</td>
</tr>
<tr>
<td>Duration (days)</td>
<td>3</td>
<td>4</td>
<td>246</td>
<td>77</td>
</tr>
<tr>
<td>Granularity (secs)</td>
<td>120</td>
<td>120</td>
<td>300</td>
<td>20</td>
</tr>
<tr>
<td>Pairwise contact frequency (per day)</td>
<td>4.6</td>
<td>6.7</td>
<td>0.024</td>
<td>0.036</td>
</tr>
</tbody>
</table>
easily accessed by other nodes in the network, which hence ensure the performance of our proposed caching scheme.

V. CACHING SCHEME

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

1) When a data source generates new data, it pushes data to the 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 the buffer conditions of nodes involved in the pushing process.

2) A requester multicasts a query to the central nodes of NCLs to pull the data, and a central node forwards the query to the nodes caching the data. A number of cached data copies are returned to the requester, and we optimize the tradeoff between data accessibility and transmission overhead by probabilistically controlling the number of returned data copies.

3) Utility-based cache replacement is conducted whenever two caching nodes contact each other, and ensures that popular data is cached nearer to the central nodes by solving a knapsack problem. 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 the NCLs by independently sending a data copy to each central node representing a NCL. Since each node in the network maintains the information about the shortest opportunistic paths to the central nodes, we use the opportunistic path weight to the central node as the relay selection metric for such data forwarding. A relay forwards data to another node with higher metric than itself, and deletes its own data copy afterwards. According to Definition 1 on opportunistic path, such strategy probabilistically ensures that each forwarding reduces the remaining delay for the data to be delivered to the corresponding central node.

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

Such determination of caching location is illustrated in Figure 5 where the solid lines indicate opportunistic contacts used to forward data, and the dashed lines indicate data forwarding stopped by the node buffer constraint. The central node C1 is able to cache the data, but the data copies being forwarded to C2 and C3 are stopped and cached at the relays R2 and R3 respectively, because neither C2 nor R3 has enough buffer to cache the data. Note that the caching location at a NCL may not have contacted the corresponding central node directly, like the case of nodes C3 and R3 in Figure 5.

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

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

4Since the data is newly generated and has not been requested yet, no cache replacement is necessary at the relay.
essentially facilitates the delivery of user queries to the caching nodes, which is described in Section V-B.

B. Queries

We assume that any node may request data, and therefore data requesters are randomly distributed in the network. A requester multicasts the query with a finite time constraint to the central nodes to pull the data. Various existing multicast schemes in DTNs [12], [19], [13] can be exploited for this purpose, and the selection of a specific multicast scheme depends on the application performance requirements.

After having received the query, a central node immediately replies to the requester with the data if it is cached locally. The data can be sent to the requester by any existing data forwarding protocol in DTNs. Otherwise, it broadcasts the query to the nodes nearby. This process is illustrated in Figure 6. 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 reply from the 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 the query history, which is then used in Section V-D for cache replacement.

C. Probabilistic Response

As shown in Figure 6, due to the probabilistic nature of data delivery in DTNs, multiple data copies are replied to the requester from NCLs to ensure that the requester is able to receive 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 the network bandwidth. The major challenge for solving this problem arises from the intermittent network connectivity in DTNs, and can be elaborated in two aspects. First, it is difficult for the caching nodes to promptly communicate with each other, and therefore the optimal number of data copies returned to the requester cannot be determined in advance by the caching nodes. Second, a relay carrying a data copy is unable to know the locations of other data copies being returned, and therefore cannot determine whether the requester has already 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 there is no tight constraint on the network storage and bandwidth, each node is able to maintain the information about the shortest opportunistic paths to all the other nodes in the network. According to Eq. (2), $C$ can determine whether to reply data to $R$ with the probability $p_{CR}(T_q - t_0)$. This probability is essentially the weight of the shortest opportunistic path from $C$ to $R$, and indicates the probability that the data can be transmitted from $C$ to $R$ within the remaining time $T_q - t_0$ for responding to the query.

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

$$p_R(t) = \frac{k_1}{1 + e^{-k_2t}},$$

where $k_1 = 2p_{\text{min}}, \; k_2 = \frac{1}{T_q} \ln \left( \frac{p_{\text{max}}}{2p_{\text{min}} - p_{\text{max}}} \right)$. The quantities $p_{\text{max}}$ and $p_{\text{min}}$ in Eq. (3) are user-specified parameters of the maximum and minimum response probabilities. As an example, the sigmoid function with $p_{\text{min}} = 0.45$, $p_{\text{max}} = 0.8$, and $T_q = 10$ hours is shown in Figure 7.

D. Cache Replacement

For each data item in the network, the locations where it is cached are dynamically adjusted via cache replacement.

\[\text{Fig. 6. Pulling data from the NCLs}\]

\[\text{Fig. 7. Probability for deciding data response}\]
This replacement is based on the current data popularity in the network, and generally places the popular data nearer to the central nodes of the NCLs. Traditional cache replacement strategies such as LRU, which removes the least-recently-used data from the cache when new data is available, are ineffective due to overly 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 either. In this paper, 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.

1) Data Popularity: The popularity of a data item in the network is probabilistically estimated based on the occurrences of the past $k$ requests to this data, which happened during the time period $[t_1, t_k]$. We assume that such occurrences of data requests in the past 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 the data expires. If data $d_i$ expires at time $t_e$, the popularity $w_i$ of $d_i$ is written as

$$w_i = 1 - e^{-\lambda_d(t_e - t_1)},$$

which is actually the probability that $d_i$ is requested at least once again in the future before time $t_e$. To calculate the popularity of a data item, 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.

2) Basic Strategy: Cache replacement occurs whenever two caching nodes $A$ and $B$ contact each other, such that the two nodes exchange their cached data to optimize the cumulative data access delay. More specifically, we collect the cached data at both nodes into a selection pool $S = \{d_1, ..., d_n\}$, and formulate cache replacement as the following knapsack problem:

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

Afterwards, node $B$ selects the data to cache from the remaining part of $S$ by solving a similar problem to Eq. (7). Since $S_A$ and $s_i$ in Eq. (7) are usually integers in numbers of bytes, this knapsack problem can be solved in pseudopolynomial time $O(n \cdot S_A)$, by using a dynamic programming approach [22].

This replacement process is illustrated by an example shown in Figure 8, where initially node $A$ caches data $d_3, d_2$ and $d_1$, 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 8(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 $B$.

Note that in cases of limited cache space, some cached data with lower popularity may be removed from the caching buffer. As shown in Figure 8(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 $B$. In this case, the data $d_6$ with the lowest popularity will be removed from the 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 8(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 $d_0$ out from the cache of node $B$ may greatly impair the accessibility of $d_0$, because there may be only few cached copies of $d_0$ 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. (7) is solved by the dynamic programming approach. More specifically, if data $d_i$ is selected by the dynamic programming algorithm, it has the probability $u_i$ to be cached at node $A$. This algorithm is described in detail in Algorithm 1, where GetMax$(S, S_A)$ calculates the maximal possible value of the items in the knapsack via dynamic programming, and SelectData$(d_{i_{\text{max}}})$ determines whether to select data $d_{i_{\text{max}}}$ to cache at node $A$ by conducting a Bernoulli experiment with the probability $u_{i_{\text{max}}}$. Note that such probabilistic selection may be iteratively conducted multiple times, in order 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.

E. Discussions

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

Hence, the data access delay is closely related with the number ($K$) of NCLs, which are represented by the top $K$ nodes with the highest NCL selection metric values in the network. 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 to be longer. Meanwhile, since the total amount of data being cached in the network is smaller, data is more likely to be cached nearer to the central nodes, and the second part of the delay will be shorter.

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. In this case, 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:

- **No Cache**, in which caching is not used for data access, and each query result is returned only by the data source.
- **Random Cache**, in which every requester caches the received data to facilitate data access in the future.
- **CacheData** [29], 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** [23], 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.

Algorithm 1: Probabilistically Data Selection at node $A$ among the data set $S$

$$i_{\text{min}} = \arg \min_i \{ s_i \mid d_i \in S, x_i = 0 \}$$

while $S \neq \emptyset \& \& S_A > s_{i_{\text{min}}}$ do

$V_{\text{max}} = \text{GetMax}(S, S_A)$

$S' = S$

while $S' \neq \emptyset \& \& V_{\text{max}} > 0$ do

$i_{\text{max}} = \arg \max_i \{ u_i \mid d_i \in S' \}$

if SelectData$(d_{i_{\text{max}}})$ = true $\& \& V_{\text{max}} \geq s_{i_{\text{max}}}$ then

$x_{i_{\text{max}}} = 1$

$S = S \setminus d_{i_{\text{max}}}$

$S_A = S_A - s_{i_{\text{max}}}$

$V_{\text{max}} = V_{\text{max}} - s_{i_{\text{max}}}$

$S' = S' \setminus d_{i_{\text{max}}}$

$i_{\text{min}} = \arg \min_i \{ s_i \mid d_i \in S, x_i = 0 \}$
A. Experiment Setup

Our performance evaluations are performed on the Info-com06 and MIT Reality traces collected from realistic DTNs, and the details of the two traces are summarized in Table 1. We assume that each node is able to communicate with other nodes in contact through bidirectional wireless links with a capacity of 2.1 Mb/s (Bluetooth EDR). In all the experiments, a node updates its contact rates with other nodes in real time based on the up-to-date contact counts since the network starts, and furthermore maintains the information about the shortest opportunistic path to the central nodes. The first half of the trace is used as the 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 the trace.

1) Data Generation: Each node in the network periodically checks whether it has generated data which has not expired yet. If not, the node determines whether to generate new data with a unified probability $p_G$. Each generated data has finite lifetime whose value is uniformly distributed in the range $[0.5T_L, 1.5T_L]$, and the period for data generation decision is also set as the average data lifetime $T_L$. For simplicity, in our evaluations we fix $p_G = 0.2$, and the amount of data in the network is hence controlled by $T_L$, as illustrated in Figure 9(a) for the MIT Reality trace.

Similarly, the data size is uniformly distributed in the range $[0.5s_{avg}, 1.5s_{avg}]$, and the caching buffer of nodes is uniformly distributed in the range $[200\text{Mb}, 600\text{Mb}]$. In the experiments, the parameter $s_{avg}$ is adjusted to simulate different node buffer conditions.

2) Query Pattern: Queries are randomly generated at all the nodes in the network, and each query is associated with a finite time constraint $T_L/2$. We assume that the query pattern follows a Zipf distribution, which has been proved to appropriately describe the query pattern of web data access [3]. More specifically, Let $P_j \in [0, 1]$ be the probability that data $j$ is requested, and $M$ be the total number of data items in the network, we have

$$P_j = \frac{1/j^s}{\sum_{i=1}^{M} 1/i^s},$$

where $s$ is an exponent parameter. The values of $P_j$ with different values of $s$ are shown in Figure 9(b). Every time $T_L/2$, each node in the network determines whether to request data $j$ with the 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 according to the trace-based validation results shown in Figure 4(c) and generate the query pattern following the Zipf distribution with exponent $s = 1$. By default, the average data lifetime $T_L$ is set as 1 week, and the average data size $s_{avg}$ is set as 100 Mb. These two parameters are then adjusted for different performance evaluation purposes.

The simulation results with different values of the average data lifetime are shown in Figure 10(a). The successful ratio of data access is mainly restrained by the data lifetime itself. As shown in Figure 10(a) when $T_L$ increases from 12 hours to 3 months, the successful ratio of all the schemes is significantly improved, because data has more time to be delivered to the requesters before expiration. In general, due to the heterogeneity of node contact pattern in DTNs, the selected NCLs are effective in communicating with other nodes in the network, and therefore our proposed intentional caching scheme achieves much better successful ratio and delay of data access. As shown in Figures 10(a) and 10(b) the performance of our scheme is 200% better than that of NoCache, and also exhibits 50% improvement over BundleCache, in which mobile nodes also incidentally cache the pass-by data. Comparatively, RandomCache is very ineffective because the requesters are randomly distributed in the network, and CacheData is also inappropriate to be used in DTNs, due to the difficulty of collecting necessary network information about query history.

Meanwhile, Figure 10(c) shows that our proposed scheme only requires moderate cache size, which is much lower than that required by RandomCache and BundleCache, especially in cases of larger $T_L$. RandomCache consumes the largest caching buffer, such that each data has 5 cached copies when $T_L$ increases to 3 months. The major reason is that each requester blindly caches any received data until its buffer is filled up. Comparatively, CacheData consumes 30% less buffer than our scheme, but also leaves a lot of data uncached, and therefore seriously impairs the data access performance.

We also evaluated the performance of data access with different node buffer conditions, which is realized by adjusting the average data size $s_{avg}$. The simulation results are shown in Figure 11(a) When the data size becomes larger, less data can be cached at the mobile nodes as shown in Figure 11(c) and therefore the performance of data access is generally reduced. In Figures 11(a) and 11(b) when $s_{avg}$ increases from 20 Mb to 200 Mb, the successful ratio of our scheme decreases from 60% to 45%, and the data access delay increases from 18 hours to 25 hours. However, the performances of other schemes even decrease much faster, and make the advantage of the performance of our scheme even larger when the 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.

![Fig. 9. Experiment setup](image-url)
C. Effectiveness of Cache Replacement

In this section, we evaluate the effectiveness of our proposed cache replacement strategy in Section V-D for improving the performance of data access. 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 the MIT Reality trace for such evaluation, and set the average data lifetime ($T_L$) as 1 week. The simulation results are shown in Figure 12. Unsurprisingly, the traditional cache replacement strategies are ineffective due to the lack of properly considering data popularity, which leads to poor data access performance. As shown in Figure 12(a), when the data size is small and the subsequent node buffer constraint is not tight, cache replacement will not be frequently conducted. Therefore, the successful ratio of the traditional strategies is only 10%-20% lower than that of our scheme. However, when the data size becomes larger, these strategies do not always select the most appropriate data to cache, and therefore the advantage of our scheme rises to over 100% when $s_{avg} = 200$ Mb. Similarly, the data access delay of these traditional strategies becomes much longer when $s_{avg}$ increases, as shown in Figure 12(b).

In Figure 12(c), we also compared the overhead of those different cache replacement strategies, which is generally the amount of data exchanged for cache replacement, and is measured by the average number for data items to be replaced before expiration. 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

The performance of data access of our scheme is tightly related with the number ($K$) of NCLs in the network. In this section, we investigate the impact of different values of $K$ on the performance of data access, using the Infocom06 trace. We set the average data lifetime $T_L = 3$ hours, and all the other parameters remain the same as in Section VI-B.

The simulation results with different node buffer conditions are shown in Figure 13. When $K$ is small, as discussed in Section V-E, it generally takes longer to forward queries and data between the requesters and caching nodes, and hence the performance of data access is reduced. This reduction is particularly significant when $K < 3$. As shown in Figures 13(a) and 13(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 the performance of data access, because the newly selected central nodes are essentially not good at communicating with other nodes in the network.

Meanwhile, as shown in Figure 13(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 the node buffer constraint is tight.

In summary, when the node buffer constraint is tight, using smaller value of $K$ is helpful to provide acceptable caching performance with lower overhead. However, the exploitation of too many NCLs will not provide any extra benefit, and may even impair the caching performance. From Figure 13 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 4(b).

VII. CONCLUSIONS

In this paper, we propose a novel scheme to support cooperative caching in DTNs, in order to provide effective data access to mobile users. Our basic idea is to intentionally cache data at a pre-specifed set of NCLs in the network, 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. Cache replacement is performed whenever two caching nodes contact each other, to ensure that popular data is cached nearer to the central nodes of NCLs. 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.

REFERENCES


