Distributed Maintenance of Cache Freshness in Opportunistic Mobile 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

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

***Abstract*—Opportunistic mobile networks consist of personal mobile devices which are intermittently connected with each other. Data access can be provided to these devices via cooperative caching without support from the cellular network infrastructure, but only limited research has been done on maintaining the freshness of cached data which may be refreshed periodically and is subject to expiration. In this paper, we propose a scheme to efficiently maintain cache freshness. Our basic idea is to let each caching node be only responsible for refreshing a specific set of caching nodes, so as to maintain cache freshness in a distributed and hierarchical manner. Probabilistic replication methods are also proposed to analytically ensure that the freshness requirements of cached data are satisfied. Extensive trace-driven simulations show that our scheme significantly improves cache freshness, and hence ensures the validity of data access provided to mobile users.**

I. Introduction

In recent years, personal hand-held mobile devices such as smartphones are capable of storing, processing and displaying various types of digital media contents including news, music, pictures or video clips. It is hence important to provide efficient data access to mobile users with such devices. Opportunistic mobile networks, which are also known as Delay Tolerant Networks (DTNs) [13] or Pocket Switched Networks (PSNs) [20], are exploited for providing such data access without support of cellular network infrastructure. In these networks, it is generally difficult to maintain end-to-end communication links among mobile users. Mobile users are only intermittently connected when they opportunistically contact, i.e., moving into the communication range of the short-range radio (e.g., Bluetooth, WiFi) of their smartphones.

Data access can be provided to mobile users via cooperative caching. More specifically, data is cached at mobile devices based on the query history, so that queries for the data in the future can be satisfied with less delay. Currently, research efforts have been focusing on determining the appropriate caching locations [27], [19], [17] or the optimal caching policies for minimizing the data access delay [28], [22].

However, there is only limited research on maintaining the freshness of cached data in the network, despite the fact that media contents may be refreshed periodically. In practice, the refreshing frequency varies according to the specific content characteristics. For example, the local weather report is usually refreshed daily, but the media news at websites of CNN or New York Times may be refreshed hourly. In such cases, the versions of cached data in the network may be out-of-date, or even be completely useless due to expiration.

The maintenance of cache freshness in opportunistic mobile networks is challenging due to the intermittent network connectivity and subsequent lack of information about cached data. First, there may be multiple data copies being cached in the network, so as to ensure timely response to user queries. Without persistent network connectivity, it is generally difficult for the data source to obtain information about the caching locations or current versions of the cached data. It is therefore challenging for the data source to determine “where to” and “how to” refresh the cached data. Second, the opportunistic network connectivity increases the uncertainty of data transmission and complicates the estimation of data transmission delay. It is therefore difficult to determine whether the cached data can be refreshed on time.

In this paper, we propose a scheme to address these challenges and to efficiently maintain freshness of the cached data. Our basic idea is to organize the caching nodes[1] as a tree structure during data access, and let each caching node be responsible for refreshing the data cached at its children in a distributed and hierarchical manner. The cache freshness is also improved when the caching nodes opportunistically contact each other. To the best of our knowledge, our work is the first which specifically focuses on cache freshness in opportunistic mobile networks.

Our detailed contributions are as follows:

- We investigate the refreshing patterns of realistic web contents. We observe that the distributions of inter-refreshing time of the RSS feeds from major news websites exhibit hybrid characteristics of exponential and power-law, which have been validated by both empirical and analytical evidences.
- Based on the experimental investigation results, we analytically measure the utility of data updates for refreshing the cached data via opportunistic node contacts. These

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.

[1] In the rest of this paper, the terms “devices” and “nodes” are used interchangeably.

utilities are calculated based on a probabilistic model to measure cache freshness. They are then used to opportunistically replicate data updates and analytically ensure that the freshness requirements of cached data can be satisfied.

The rest of this paper is organized as follows. In Section II we briefly review the existing work. Section III provides an overview about the models and caching scenario we use, and also highlights our basic idea. Section IV presents our experimental investigation results on the refreshing patterns of real web sites. Sections V and VI describe the details of our proposed cache refreshing schemes. The results of trace-driven performance evaluations are shown in Section VII, and Section VIII concludes the paper.

II. Related Work

Due to the intermittent network connectivity in opportunistic mobile networks, data is forwarded in a "carry-and-forward" manner. Node mobility is exploited to let nodes physically carry data as relays, and forward data opportunistically when contacting others. The key problem is hence how to select the most appropriate nodes as relays, based on the prediction of node contacts in the future. Some forwarding schemes do such prediction based on node mobility patterns [9], [33], [14]. In some other schemes [4], [1], stochastic node contact process is exploited for better prediction accuracy. Social contact patterns of mobile users, such as centrality and community structures, have also been exploited for relay selection [10], [21], [18].

Based on this opportunistic communication paradigm, data access can be provided to mobile users in various ways. In some schemes [23], [16], data is actively disseminated to specific users based on their interest profiles. Publish/subscribe systems [32], [24] are also used for data dissemination by exploiting social community structures to determine the brokers.

Caching is another way to provide data access. Determining appropriate caching policies in opportunistic mobile networks is complicated by the lack of global network information. Some research efforts focus on improving data accessibility from infrastructure networks such as WiFi [19] or Internet [27], and some others study peer-to-peer data sharing among mobile nodes. In [17], data is cached at specific nodes which can be easily accessed by others. In [28], [22], caching policies are dynamically determined based on data importance, so that the aggregate utility of mobile nodes can be maximized.

When the versions of cached data in the network are heterogeneous and different from that of the source data, research efforts have been focusing on maintaining the consistency of these cache versions [7], [11], [5], [6]. Being different from existing work, in this paper we focus on ensuring the freshness of cached data, i.e., the version of any cached data should be as close to that of the source data as possible. [22] discussed the practical scenario in which data is periodically refreshed, but did not provided specific solutions for maintaining cache freshness. We propose methods to maintain cache freshness in a distributed and hierarchical manner, and analytically ensure that the freshness requirement of cached data can be satisfied.

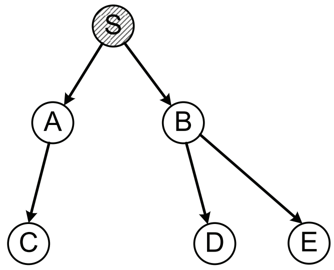


Fig. 1. Data Access Tree (DAT). Each node in the DAT accesses data when it contacts its parent node in the DAT.

III. Overview

A. Models

*1) **Network Model**:* Opportunistic contacts among nodes are described by a network *contact graph* $G(V, E)$, where the contact process between a node pair $i, j \in V$ is modeled as an edge $e_{ij} \in E$. The characteristics of an edge $e_{ij} \in E$ are determined by the properties of inter-contact time among nodes. Similar to previous work [1], [34], we consider the pairwise node inter-contact time as exponentially distributed. Contacts between nodes i and j then form a Poisson process with contact rate λ_{ij}, which is calculated in real time from the cumulative contacts between nodes i and j.

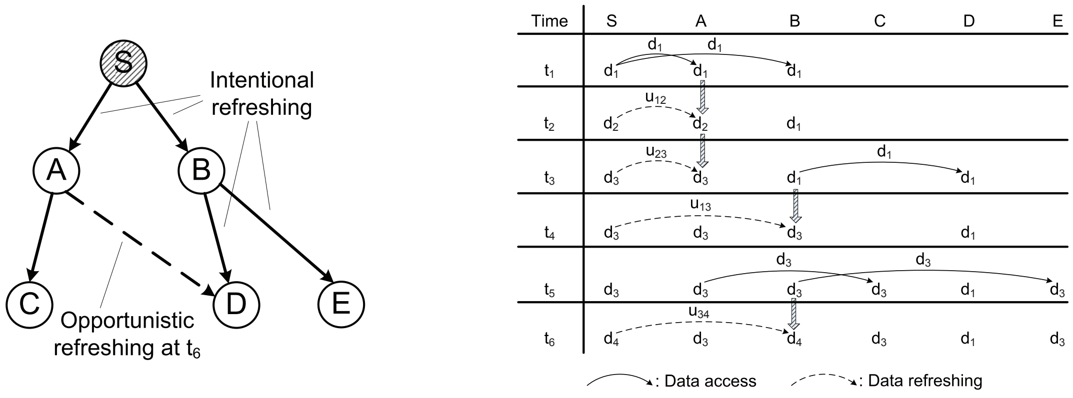
*2) **Cache Freshness Model**:* We focus on ensuring the freshness of cached data, i.e., the version of any cached data should be as close to that of the source data as possible. Letting v_S^t denote the version number of source data at time t and v_j^t denote that of data cached at node j, our requirement on cache freshness is probabilistically described as

$$\mathbb{P}(v_j^t \geq v_S^{t-\Delta}) \geq p, \qquad (1)$$

for any time t and any node j. The version number is initialized as 0 when data is first generated and monotonically increased by 1 every time the data is refreshed.

Higher network storage and transmission overhead is generally required for decreasing Δ or increasing p. Hence, our proposed model provides the flexibility to tradeoff between cache freshness and network maintenance overhead according to the specific data characteristics and applications. For example, news from CNN or the New York Times may be refreshed frequently, and smaller Δ (e.g., 1 hour) should be applied accordingly. In contrast, the local weather report may be updated daily, and the requirement on Δ can hence be relaxed to avoid unnecessary network cost. The value of p may be flexible based on user interests in the data. However, there are cases where an application might have specific requirements on Δ and p to achieve sufficient levels of data freshness.

*3) **Data Update Model**:* Whenever data is refreshed, the data source computes the difference between the current and previous versions and generates a data update. Cached data is refreshed by such update instead of complete data for better storage and transmission efficiency. This technique is called *Delta encoding*, which has been applied in web caching for reducing Internet traffic [26].



(a) Intentional and opportunistic refreshing (b) Temporal sequence of data access and refreshing operations

Fig. 2. Distributed and hierarchical maintenance of cache freshness

Letting u_{ij} denote the update of data from version i to version j, we assume that any caching node is able to refresh the cached data as $d_i \otimes u_{ij} \rightarrow d_j$, where d_i and d_j denote the data with version i and j, respectively. We also assume that any node is able to compute u_{ij} from d_i and d_j.

When data has been refreshed multiple times, various updates for the same data may co-exist in the network. We assume that any node is able to merge consecutive data updates, i.e., $u_{ij} \oplus u_{jk} \rightarrow u_{ik}$. However, d_j cannot be refreshed to d_k by u_{ik} even if $j > i$. For example, u_{14} which is produced by merging u_{13} and u_{34} cannot be used to refresh d_3 to d_4.

B. Caching Scenario

Mobile nodes share data generated by themselves or obtained from the Internet. In this paper, we consider a generic caching scenario which is also used in [22]. The query generated by a node is satisfied as soon as this node contacts some other node caching the data. During the mean time, the query is stored at the requesting node. After the query is satisfied, the requesting node caches the data locally for answering possible queries in the future. Each cached data item is associated with a finite lifetime and is automatically removed from cache when it expires. The data lifetime may change each time the cached data is refreshed.

In practice, when multiple data items with varied popularity compete for the limited buffer of caching nodes, more popular data is prioritized to ensure that the cumulative data access delay is minimized. Such prioritization is generally formulated as a knapsack problem [17] and can be solved in pseudo-polynomial time using a dynamic programming approach [25]. Hence, the rest of this paper will focus on ensuring the freshness of cached copies of a specific data item. The consideration of multiple data items and limited node buffer is orthogonal to the major focus of this paper.

In the above scenario, data is essentially disseminated among nodes interested in the data when they contact each other, and these nodes form a "Data Access Tree (DAT)" as shown in Figure 1. Queries of nodes A and B are satisfied when they contact the data source S. Data cached at A and B are then used for satisfying queries from nodes C, D and E.

Due to intermittent network connectivity, each node in the DAT only has knowledge about data cached at its children. For example, after having its query satisfied by S, A may lose its connection with S due to mobility, and hence A is unaware of the data cached at nodes B, D and E. Similarly, S may only be aware of data cached at nodes A and B. Such limitation makes it challenging to maintain cache freshness, because it is difficult for the data source to determine "where to" and "how to" refresh the cached data.

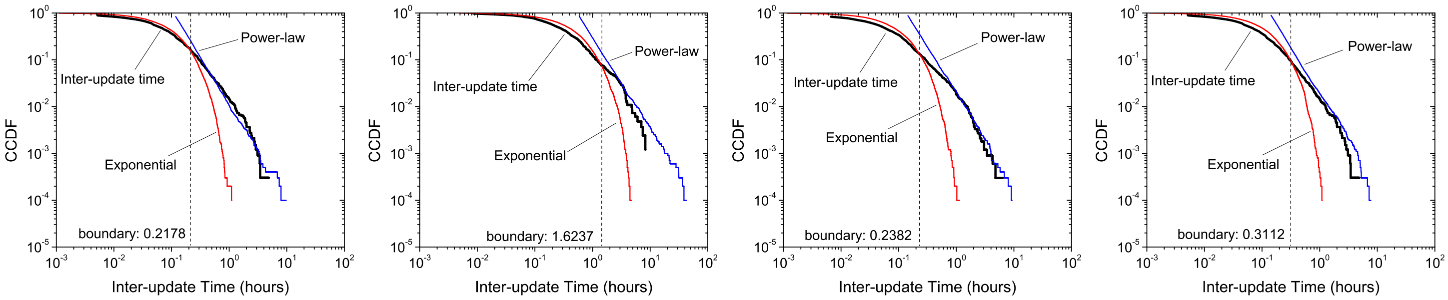
C. Basic Idea

Our basic idea for maintaining cache freshness is to refresh the cached data in a distributed and hierarchical manner. As illustrated in Figure 2, this refreshing process is split into two parts, i.e., the intentional refreshing and the opportunistic refreshing, according to whether the refreshing node has the knowledge about the cached data to be refreshed.

In intentional refreshing, each node is only responsible for refreshing data cached at its children in the DAT. For example, in Figure 2(a) node S is only responsible for refreshing data cached at A and B. Since A and B obtain their cached data from S, S has knowledge about the versions of their cached data and is able to prepare the appropriate data updates accordingly. In the example shown in Figure 2(b), S refreshes data cached at A and B using updates u_{23} and u_{13}, when S contacts A and B at time t_3 and t_4 respectively. In Section V, these updates are also opportunistically replicated to ensure that they can be delivered to A and B on time. Particularly, the topology of DAT may change due to the expiration of cached data. When A is removed from the DAT due to cache expiration, its child C only re-connects to the DAT and gets updated when C contacts another node in the DAT.

In opportunistic refreshing, a node refreshes any cached data with older versions whenever possible upon opportunistic contact. For example in Figure 2(a), when node A contacts node D at time t_6, A updates the data cached at D from d_1 to d_3. Since A does not know the version of the data cached at D, it cannot prepare u_{13} for D in advance[2]. Instead, A has to transmit the complete data d_3 to D with

[2]The update u_{13} can only be calculated using d_1 and d_3.



(a) CNN Top Stories (b) BBC Politics (c) NYTimes Sports (d) Business Week Daily

Fig. 3. CCDF of inter-refreshing time of individual RSS feeds

No.	RSS feed	Number of updates	Avg. inter-refreshing time (hours)
1	CNN Top Stories	2051	0.2159
2	NYTimes US	4545	0.0954
3	CNN Politics	623	0.7166
4	BBC Politics	827	0.5429
5	ESPN Sports	2379	0.1856
6	NYTimes Sports	3344	0.1355
7	Business Week Daily	4783	0.0948
8	Google News Business	7266	0.061
9	Weather.com NYC	555	0.8247
10	Google News ShowBiz	5483	0.0808
11	BBC ShowBiz	531	0.8506

TABLE I
NEWS UPDATES RETRIEVED FROM WEB RSS FEEDS

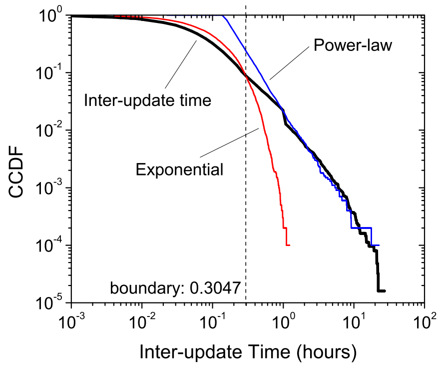


Fig. 4. Aggregate CCDF of the inter-refreshing time in log-log scale

higher transmission overhead. In Section VI, we propose to probabilistically determine whether to transmit the complete data according to the chance of satisfying the requirement of cache freshness, so as to optimize the tradeoff between cache freshness and network transmission overhead.

IV. Refreshing Patterns of Web Contents

In this section, we investigate the refreshing patterns of realistic web contents, as well as their temporal variations during different time periods in a day. These patterns highlight the homogeneity of data refreshing behaviors among different data sources and categories, and suggest appropriate calculation of utilities of data updates for refreshing cached data.

A. Datasets

We investigate the refreshing patterns of categorized web news. We dynamically retrieved news updates from news websites including CNN, New York Times, BBC, Google News, etc, by subscribing to their public RSS feeds. During the 3-week experiment period between 10/3/2011 and 10/21/2011, we have retrieved a total number of 32787 RSS updates from 11 RSS feeds in 7 news categories. The information about these RSS feeds and retrieved news updates is summarized in Table I, which shows that the RSS feeds differ in their numbers of updates and the update frequencies.

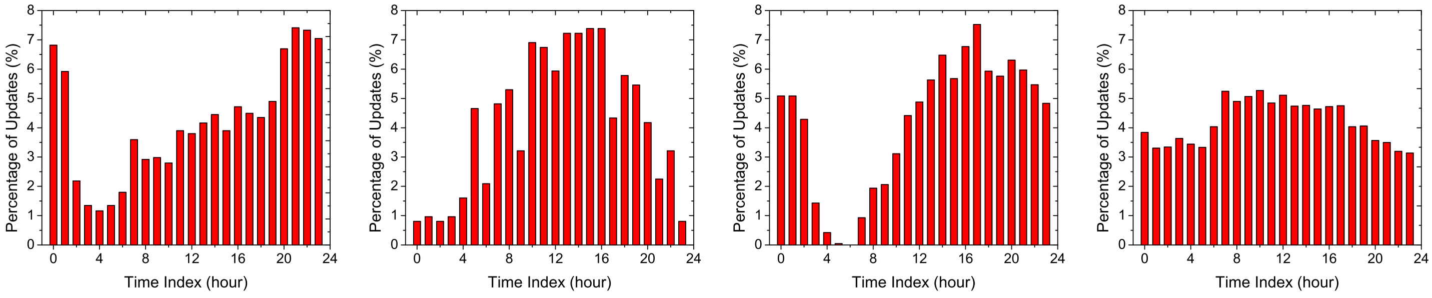
B. Distribution of Inter-Refreshing Time

We provide both empirical and analytical evidence of a dichotomy in the Complementary Cumulative Distribution Function (CCDF) of the inter-refreshing time, which is defined as the time interval between two consecutive news updates from the same RSS feed. Our results show that up to a boundary on the order of several minutes, the decay of the CCDF is well approximated as exponential. In contrast, the decay exhibits power-law characteristics beyond this boundary.

*1) **Aggregate distribution**:* Figure 4 shows the aggregate CCDF of inter-refreshing time for all the RSS feeds, in log-log scale. The CCDF values exhibit slow decay over the range spanning from a few seconds to 0.3047 hour. It suggests that around 90% of inter-refreshing time falls into this range and follows an exponential distribution. Figure 4 also shows that the CCDF values of inter-refreshing time within this range is accurately approximated by the random samples drawn from an exponential distribution with the average inter-refreshing time (0.1517 hours) as parameter.

For the remaining 10% of inter-refreshing time with values larger than the boundary, the CCDF values exhibit linear decay which suggests a power-law tail. To better examine such tail characteristics, we also plot the CCDF of a generalized Pareto distribution with the shape parameter $\xi = 0.5$, location parameter $\mu = 0.1517$ and scale parameter $\sigma = \mu \cdot \xi = 0.0759$. As shown in Figure 4, the Pareto CCDF closely approximates that of the inter-refreshing time beyond the boundary. Especially when inter-refreshing time is longer than 1 hour, the two curves almost overlap with each other.

*2) **Distributions of individual RSS feeds**:* Surprisingly, we found that the distributions of inter-refreshing time of individual RSS feeds exhibit similar characteristics with that of the aggregate distribution. For example, for the two RSS



(a) NYTimes US (b) CNN Politics (c) ESPN Sports (d) Google News Business

Fig. 5. Temporal distribution of news updates during different hours in a day

No. RSS feed	Boundary (hours)	Exponential		generalized Pareto	
		percent. of updates (%)	α (%)	percent. of updates (%)	α (%)
1	0.2178	91.07	4.33	9.93	5.37
2	0.3245	84.24	6.71	15.76	3.28
3	1.9483	88.12	7.24	11.88	3.65
4	1.6237	86.75	5.69	13.25	4.45
5	0.2382	93.37	6.54	6.63	4.87
6	0.2754	92.28	6.73	7.72	2.12
7	0.3112	87.63	5.26	12.37	3.13
8	0.2466	89.37	8.45	10.63	2.64
9	1.7928	90.22	11.62	9.78	8.25
10	0.1928	88.57	6.75	11.43	3.58
11	2.0983	83.32	7.44	16.68	3.23

TABLE II
NUMERICAL RESULTS FOR DISTRIBUTIONS OF INTER-REFRESHING TIME OF INDIVIDUAL RSS FEEDS

feeds in Figure 3 with different news categories, the CCDF decay of each RSS feed is analogous to that of the aggregate CCDF in Figure 4. Figure 3 shows that the boundaries for different RSS feeds are heterogeneous and mainly determined by the average inter-refreshing time. These boundaries are summarized in Table II.

To quantitatively justify the characteristics of exponential and power-law decay in the CCDF of individual RSS feeds, we perform a Kolmogorov-Smirnov goodness-of-fit test [30] on each of the 11 RSS feeds listed in Table I. For each RSS feed, we collect the inter-contact times smaller than its boundary and test whether the null hypothesis "these inter-contact times are exponentially distributed" can be accepted. A similar test is performed on the inter-contact times with larger values for the generalized Pareto distribution.

The significance levels (α) for these null hypotheses being accepted are listed in Table II. The lower the significance level is, the more confident we are that the corresponding hypothesis is statistically true. As shown in Table II, for all the RSS feeds, the probability for erroneously accepting the null hypotheses is lower than 10%, which is the significance level usually being used for statistical hypothesis testing [8]. Particularly, the significance levels for accepting a generalized Pareto distribution are generally better than those for accepting an exponential distribution.

C. Temporal Variations

We are also interested in the temporal variations of the RSS feeds' updating patterns. Figure 5 shows the temporal distribution of news updates from RSS feeds over different hours in a day. We observe that the characteristics of such temporal variation are heterogeneous with different RSS feeds. For example, the majority of news updates from NYTimes and ESPN are generated during the time period from the afternoon to the evening. Comparatively, the news updates from Google News are evenly distributed among different hours in a day.

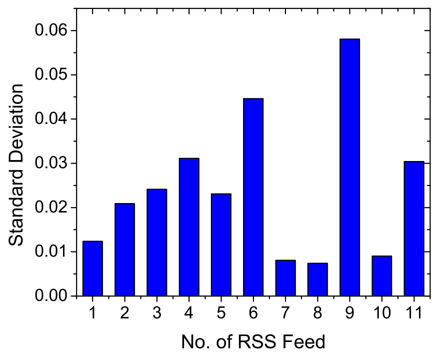


Fig. 6. Standard deviation of the numbers of news updates during different hours in a day

To better quantify the skewness of such temporal variation, we calculate the standard deviation of the numbers of news updates during different hours in a day for each of the 11 RSS feeds listed in Table I, and the calculation results are shown in Figure 6. By comparing Figure 6 with Figure 5, we conclude that the temporal distributions of news updates from most RSS feeds are highly skewed. The transient distribution of inter-refreshing time of a RSS feed during specific time periods hence may differ a lot from its cumulative distribution. Such temporal variation may affect the performance of maintaining cache freshness, and will be evaluated in detail via trace-driven simulations in Section VII.

V. Intentional Refreshing

In this section, we explain how to ensure that data updates are delivered to the caching nodes on time, so that the freshness requirements of cached data are satisfied. Based on investigation results on the distribution of inter-refreshing time in Section IV, we calculate the utility of each update which estimates the chance for the requirement being satisfied by this update. Such utility is then used for opportunistic replication of data updates.

Notation	Explanation
t_C	Current time when B generates data update for D
t_L	The last time when B contacted D
T_u	Random variable indicating the inter-refreshing time of source data
t_0	The last time when the source data was refreshed
λ_C	Pairwise contact rate between nodes B and D

TABLE III
NOTATION SUMMARY

A. Utility of Data Updates

In practice, the requirement of cache freshness may not be satisfied due to the limited nodes' contact capability. When a node B in the DAT maintains the data update for its child D, it calculates the utility of this update which is equal to the probability that this update carried by B satisfies the freshness requirement for data cached at D. With respect to the notations in Table III which are used throughout the rest of this paper, such utility can generally be calculated as follows:

$$U = \int_{t_C}^{\infty} \mathbb{P}(B \text{ contacts } D \text{ at time } t) \cdot \mathbb{P}(T_u \geq t - t_0 - \Delta) dt. \quad (2)$$

As illustrated in Figure 7, for the utility of update u_{13} at node B, Eq. (2) measures the probability that the source data has not been refreshed to d_4 at time $t - \Delta$, given that B contacts D at time t and refreshes the data cached at D from d_1 to d_3 using u_{13}. According to our network modeling in Section III-A, Eq. (2) can be rewritten as

$$U = \int_{t_C}^{\infty} \lambda_C e^{-\lambda_C (t - t_L)} \cdot (1 - F_u(t - t_0 - \Delta)) dt, \quad (3)$$

where $F_u(t)$ is the CDF of the inter-refreshing time T_u of source data.

In general, a node in the DAT may be responsible for refreshing data cached at its k children in the DAT. In this case, the cumulative utility of data update is calculated as

$$U = 1 - \prod_{i=1}^{k} (1 - U_i),$$

where U_i is the utility of data update for the i-th child node.

The utility of data update depends on the characteristics of $F_u(t)$. In practice, each node in the DAT individually maintains the information about $F_u(t)$ based on the past history about the source data being refreshed. According to the experimental results in Section IV, we calculate the utility of data update with different distributions of T_u.

*1) **Exponential Distribution**:* For exponential distribution, we have $F_u(t) = 1 - e^{-\lambda_u t}$ where λ_u is the frequency of the source data being refreshed. Using Eq. (3), we have

$$\begin{aligned} U &= \int_{t_c}^{\infty} \lambda_C e^{-\lambda_C (t - t_L)} \cdot e^{-\lambda_u (t - t_2)} dt \\ &= \frac{\lambda_C}{\lambda_C + \lambda_u} \cdot e^{-\lambda_C (t_C - t_L)} \cdot e^{-\lambda_u (t_C - t_2)}, \end{aligned} \quad (4)$$

where $t_2 = t_0 + \Delta$.

*2) **Pareto Distribution**:* Pareto distribution exhibits power-law characteristics, and we have

$$F_u(t) = \begin{cases} 0, & \text{if } t < T_{\min} \\ 1 - (\frac{T_{\min}}{t})^{\alpha}, & \text{if } t \geq T_{\min} \end{cases}.$$

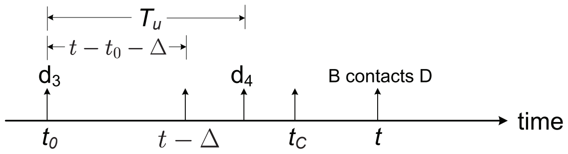


Fig. 7. Calculating the utility of data updates

We consider that $\alpha > 1$ so that the distribution has a finite mean value. As a result, if $t_C \geq T_{\min} + t_2$,

$$\begin{aligned} U &= \int_{t_C}^{\infty} \lambda_C e^{-\lambda_C (t - t_L)} \cdot (\frac{T_{\min}}{t - t_2})^{\alpha} dt \\ &= (\lambda_C T_{\min})^{\alpha} \cdot e^{-\lambda_C (t_2 - t_L)} \cdot \Gamma(1 - \alpha, \lambda_C (t_C - t_2)), \end{aligned} \quad (5)$$

where $\Gamma(s, x) = \int_x^{\infty} t^{s-1} e^{-t} dt$ is the incomplete Gamma function. Otherwise if $t_C < T_{\min} + t_2$,

$$\begin{aligned} U = & \int_{T_{\min}+t_2}^{\infty} \lambda_C e^{-\lambda_C (t - t_L)} \cdot (\frac{T_{\min}}{t - t_2})^{\alpha} dt \\ & + \int_{t_c}^{T_{\min}+t_2} \lambda_C e^{-\lambda_C (t - t_L)} dt \\ = & (\lambda_C T_{\min})^{\alpha} \cdot e^{-\lambda_C (t_2 - t_L)} \cdot \Gamma(1 - \alpha, \lambda_C T_{\min}) \\ & + e^{-\lambda_C (t_C - t_L)} - e^{-\lambda_C (T_{\min} + t_2 - t_L)} \end{aligned} \quad (6)$$

From Section IV we know that the distribution of T_u exhibits hybrid characteristics of exponential and power-law distributions. Hence, the method for utility calculation should be adaptively determined. According to Eq. (3), the utility should be calculated following Eq. (4) when the value of $t - t_0 - \Delta$ is small. Otherwise, it should be calculated following Eqs. (5) and (6).

B. Opportunistic Replication of Data Updates

If a node in the DAT finds out that the utility of the data update it carries is lower than the required probability p for maintaining cache freshness, it opportunistically replicates the data update to other nodes outside of the DAT. These nodes then act as relays to carry the data update and help deliver the update to the caching node to be refreshed.

Such a replication process is illustrated in Figure 8. Whenever node S contacts another node R_k outside of the DAT, it determines whether to replicate the data update for refreshing B to R_k. In order for S to make such a decision, R_i also calculates its utility U_{R_k} of the data update. S only replicates the data update to R_k if $U_{R_k} \geq U_{R_j}$ for $\forall j \in [0, k)$, and S itself is considered as R_0. The replication when the utilities of data update at the k selected relays satisfy

$$1 - \prod_{i=0}^{k} (1 - U_{R_i}) \geq p, \quad (7)$$

i.e., the probability that the requirement of cache freshness at B is satisfied by at least one relay is equal to or larger than p. Note that the selected relays are only able to refresh the specific data cached in the DAT, but are unable to provide data access to other nodes outside of the DAT.

Such a replication strategy is similar to Delegation [12] which focuses on forwarding data to a specific destination. Since each selected relay has a higher utility of data update

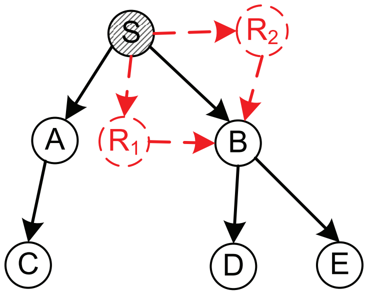


Fig. 8. Opportunistic replication of data updates

than that of all the previous relays, such a strategy efficiently ensures that the requirement of cache freshness can be satisfied without unnecessarily consuming network resources. Particularly, when the node contact frequency in the network is extremely low, S may be unable to replicate the data update to a sufficient number of relays before the source data is refreshed. In this case, the cache freshness is maintained with best-effort but the required probability p for cache freshness may not be satisfied.

VI. Opportunistic Refreshing

In addition to intentionally refreshing data cached at its children in the DAT, a node also refreshes other cached data with older versions whenever possible upon opportunistic contacts. In this section, we propose a probabilistic approach to efficiently make cache refreshing decisions and optimize the tradeoff between cache freshness and network transmission overhead.

A. Probabilistic Decision

Opportunistic refreshing is generally more expensive because the complete data usually needs to be transmitted, and its size is much larger than that of data update. As a result, it is important to make appropriate decisions on opportunistic refreshing, so as to optimize the tradeoff between cache freshness and network transmission overhead, and to avoid inefficient consumption of network resources.

We propose a probabilistic approach to efficiently refresh the cache data, and the data is only refreshed if its required freshness cannot be satisfied by intentional refreshing. In our approach, when a node A contacts node D which caches an older version of the data at time t_C, A probabilistically determines whether to transmit the complete data to D for refreshing the data cached at D. This probability p_T estimates the chance that the required freshness of D's cached data cannot be satisfied by D's parent B in the DAT via intentional refreshing, but can be satisfied if the complete data is transmitted by A. More specifically,

$$p_T = (1 - F_u(t_C - \Delta - t_0)) \cdot (1 - U_{BD}(t_C)), \tag{8}$$

where $U_{BD}(t_C)$ is the utility of data update carried by B for intentionally refreshing the data cached at D. According to Section V-A, the utility of data update is only determined by the pairwise node contact rate and the CDF $F_u(t)$ of the inter-refreshing time of the source data. Hence, $U_{BD}(t_C)$ can be calculated by D and is available to A when A contacts D. Since additional relays may be used for delivering data updates in intentional refreshing as described in Section V-B, the utility $U_{BD}(t_C)$ calculated by D essentially provides a lower bound on the actual effectiveness of intentional refreshing.

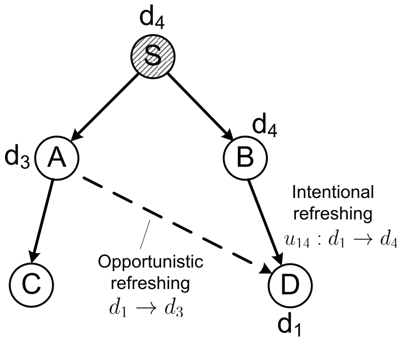


Fig. 9. Side-effect of opportunistic refreshing

B. Side-Effect of Opportunistic Refreshing

Due to possible version inconsistency among different data copies cached in the DAT, opportunistic refreshing may have some side-effects on cache freshness. Such side-effect is illustrated in Figure 9. When A opportunistically contacts node D and refreshes D's cached data from d_1 to d_3, it is unaware of the data cached at B with a newer version d_4. Meanwhile, B only knows that D caches data d_1 and prepares update u_{14} for D. As a result, when B contacts D later, it cannot update D's cached data from d_3 to d_4 using u_{14}.

Complete elimination of such side-effect is challenging due to the difficulty of timely coordination among disconnected caching nodes. Instead, we propose to let node D probabilistically estimate the chance for such side-effect to happen before its cached data is refreshed by node A at time t_C. For such estimation, each node A in the DAT maintains its opportunistic path to the data source S, which is defined as follows:

Definition 1: Opportunistic path

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

An opportunistic path is illustrated in Figure 10, and such paths can be maintained among nodes in the DAT along with the data dissemination process. Inter-contact time X_k between nodes N_k and N_{k+1} on P_{AS}, 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 time needed to transmit data from A to S is $Y = \sum_{k=1}^{r} X_k$ following a hypoexponential distribution [29], such that

$$p_Y(x) = \sum_{k=1}^{r} C_k^{(r)} p_{X_k}(x), \tag{9}$$

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

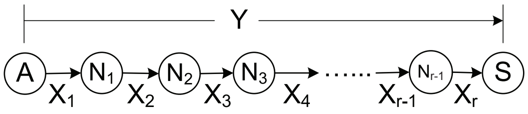


Fig. 10. Opportunistic path

From Eq. (9), the path weight is written as

$$w_{AS}(T) = \int_0^T p_Y(x)dx = \sum_{k=1}^{r} C_k^{(r)} \cdot (1 - e^{-\lambda_k T}), \quad (10)$$

which measures the data transmission delay between two nodes A and S along this path.

Essentially, such side-effect happens if the data cached at node B has been refreshed to a version newer than that of A by time t_C. This probability can be estimated as

$$p_s = \int_{t_0}^{t_C} p_u(t - t_0) \cdot w_{BS}(t_C - t)dt, \quad (11)$$

where $p_u(t) = \frac{dF_u(t)}{dt}$ is the Probability Density Function (PDF) of the refreshing interval of source data, and $w_{BS}(t_C - t)$ is the weight of opportunistic path between B and S which is defined in Eq. (10), which measures the probability that data update generated by S at time t is transmitted to B before time t_C. In practice, p_s can be applied to A's probabilistic refreshing decision when A contacts D, such that $\widetilde{p}_T = p_T \cdot p_s$.

VII. Performance Evaluations

In this section, we compare the performance of our proposed cache refreshing scheme with the following schemes:

- **Passive Refreshing**: a caching node only refreshes data cached at another node upon contact. It is different from our opportunistic refreshing scheme in Section VI in that it does not consider the tradeoff between cache freshness and network transmission overhead.
- **Active Refreshing**: every time when the source updates data, it actively disseminates the date update to the whole network. Since the data source does not maintain any information regarding the caching nodes, such dissemination is generally realized via Epidemic routing [31].
- **Publish/Subscribe**: As described in [32], [24], cached data is refreshed via the dedicated broker nodes, which receive the up-to-date data from the data source. Each broker maintains information about the caching nodes which retrieve data from this broker, and is responsible for refreshing these caching nodes upon opportunistic contacts. It is different from our scheme in that it only comprises a two-level cache refreshing hierarchy.

The following metrics are used for evaluations. Each simulation is repeated multiple times with random data sources and user queries for statistical convergence.

- **Refreshing Ratio**, the percentage of data updates that are delivered to caching nodes before the cached data expires.
- **Refreshing Delay**, the average delay for a data update being delivered from the data source to the caching nodes.
- **Refreshing overhead**: the average number of times that a data update is forwarded. Particularly, when a data item having been refreshed k times is forwarded during opportunistic refreshing, it is equivalent that a corresponding data update has been forwarded k times.

Trace	DieselNet	Infocom
Duration (days)	20	4
No. of devices	40	78
No. of internal contacts	3,268	182,951
No. of internal contacts/pair/day	0.102	7.52

TABLE IV
TRACE SUMMARY

A. Simulation Setup

Our evaluations are conducted on two realistic opportunistic mobile network traces, which record contacts among users carrying Bluetooth-enabled mobile devices. These devices periodically detect their peers nearby, and a contact is recorded when two devices move close to each other. The traces cover various types of mobile network environments including suburban areas (DieselNet [2]) and conference site (Infocom [21]). As summarized in Table IV, they differ in their network scale, duration and node contact frequency.

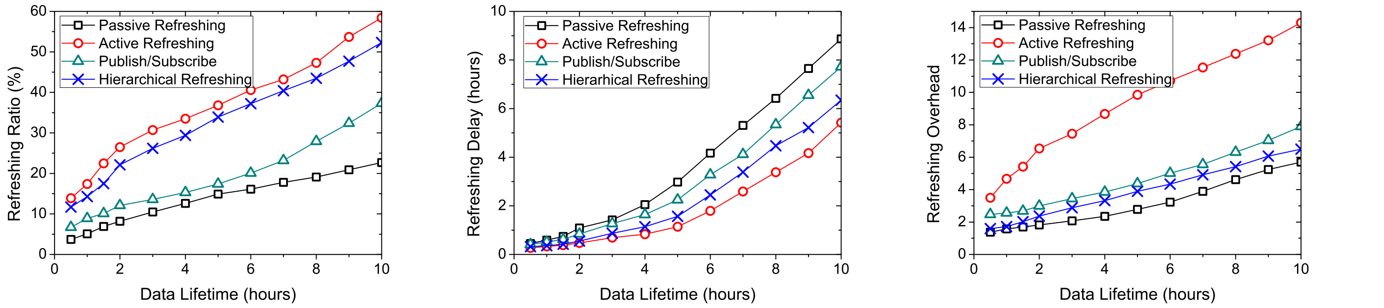
The performance of our proposed schemes is evaluated under the generic caching scenario described in Section III-B. The datasets described in Section IV are exploited to simulate the data being cached in the network, as well as the inter-refreshing time of data. Since the pairwise node contact frequency is generally lower than the data refreshing frequency, we pick up the 4 RSS feeds listed in Table I with average inter-refreshing time longer than 0.5 hours for our evaluations. For each RSS feed, a random node is selected as the data source for distributing data updates to the caching nodes.

Queries are randomly generated at all nodes, and each query has a finite time constraint T. We assume that the query pattern follows a Zipf distribution which has been widely used for modelling web data access [3]. Let $P_j \in [0, 1]$ be the probability that data j is requested, and $M = 4$ 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. Every time T, each node determines whether to request data j with probability P_j.

B. Performance of Maintaining Cache Freshness

We first compare the performance of our proposed hierarchical refreshing scheme with other schemes by varying the lifetime (L) of the cached data. For our scheme, we set $\Delta = 1.5$ hours and $p = 60\%$. The time constraint (T) of user queries is set as 5 hours.

The evaluation results are shown in Figure 11. When L increases, there is a larger amount of data being cached, and data updates have higher chances to be delivered to the caching nodes before data expiration. As shown in Figure 11(a), when L increases from 1 hour to 10 hours, the refreshing ratio has been improved by over 400%, and the average refreshing delay and overhead increase accordingly. In all cases, the performance of our scheme in refreshing ratio and delay significantly improves upon that of Passive Refreshing and Publish/Subscribe. In Figure 11(a), such advantage in



(a) Refreshing Ratio (b) Refreshing Delay (c) Refreshing Overhead

Fig. 11. Performance of maintaining cache freshness with varied lifetime of cached data

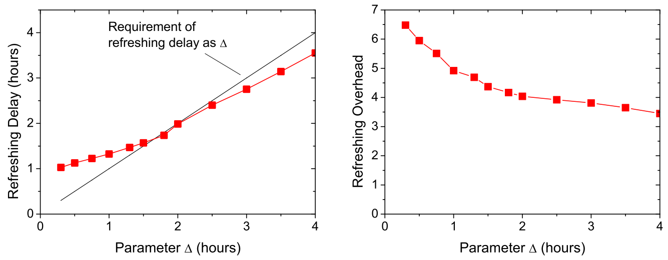
refreshing ratio can be up to 35% when L is large, and our scheme ensures that over 50% of data updates can be delivered on time when $L = 10$ hours. Similarly, a 20% reduction in refreshing delay can be achieved as shown in Figure 11(b). Active Refreshing outperforms our scheme by 10%-15%, but Figure 11(c) shows that such performance is achieved at the cost of much higher refreshing overhead.

We also evaluate the performance of our scheme with different requirements of cache freshness specified by parameters Δ and p. The parameter values are set by default as $\Delta = 1.5$ hours and $p = 60\%$, and are varied during different simulations. Data lifetime is set as $L = 5$ hours. The evaluation results are shown in Figures 12 and 13. From Figure 12 we observe that, when the value of Δ is small, the cache freshness is mainly constrained by the network contact capability, and the actual refreshing delay is much higher than the required Δ. Such inability to satisfy the cache freshness requirements leads to more replications of data updates as described in Section V-B, and makes caching nodes more prone to perform opportunistic refreshing. As a result, it increases the refreshing overhead as shown in Figure 12(b). When Δ is small, further decreasing Δ does not help on reducing refreshing delay, but unnecessarily wastes network resources on replicating data updates. Based on Figure 12, we suggest to set $\Delta \geq 2$ hours for the Infocom trace.

Figure 13 shows that increasing p helps reduce the average refreshing delay by 25%. As described in Section V-B, increasing p stimulates the caching nodes to replicate data updates, and hence increases the refreshing overhead as shown in Figure 13(b). However, since different values of p do not affect the calculation of utilities of data updates, such increase of refreshing overhead is relatively smaller than that of decreasing Δ.

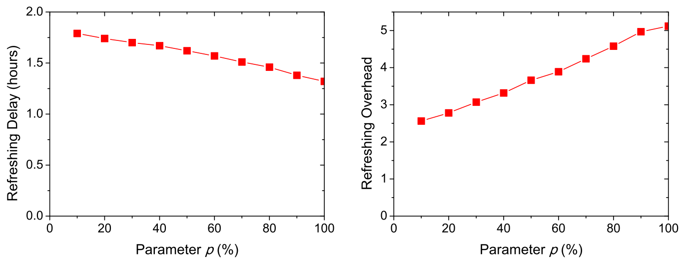
C. Temporal Variations

Section IV-C shows that the refreshing patterns of web RSS data is temporally skewed, such that the majority of data updates are generated during specific time periods of a day. Similar skewness has also been found in the transient distribution of node contacts [15]. It is therefore interesting to evaluate the temporal variation of the performance of maintaining cache freshness, which is determined by both node contact and data refreshing patterns.



(a) Refreshing delay (b) Refreshing overhead

Fig. 12. Performance of maintaining cache freshness with different values of parameter Δ and $p = 60\%$

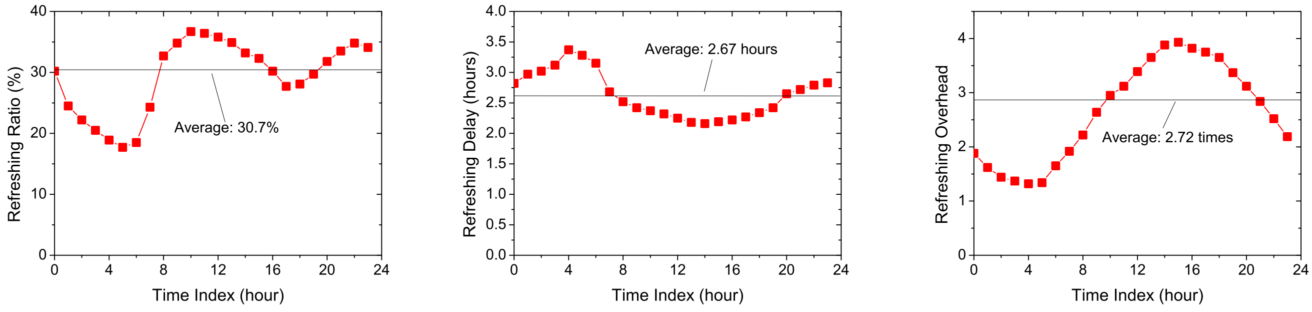


(a) Refreshing delay (b) Refreshing overhead

Fig. 13. Performance of maintaining cache freshness with different values of parameter p and $\Delta = 1.5$ hours

In this section, we evaluate such temporal variation on the DieselNet trace. Since the pairwise contact frequency in the DieselNet trace is generally low, we set $\Delta = 4$ hours and $p = 50\%$. The data lifetime L is set as 10 hours. The evaluation results are shown in Figure 14, in which the performance of maintaining cache freshness during different hours in a day is recorded separately. In general, the temporal skewness can be found in all three evaluation metrics, and is determined by the temporal distributions of both node contacts and data updates available during different hours in a day. As shown in Figure 14(a), the refreshing ratio during the time period between 8AM and 4PM is generally higher than the average refreshing ratio, because majority of node contacts have been generated during this time period according to [15]. Correspondingly, the refreshing ratio is much lower than the average level during the time period between 12AM and 8AM.

Similar trends are also found in Figures 14(b) and 14(c); the abundance of node contacts help reduce the refreshing delay but produces more data updates. In contrast, both the refresh-



(a) Refreshing Ratio (b) Refreshing Delay (c) Refreshing Overhead

Fig. 14. Temporal variations of the performance of maintaining cache freshness

ing ratio and overhead are much lower between 12AM and 8AM, when few node contacts and data updates are observed. In summary, we conclude that the transient performance of maintaining cache freshness differs a lot from the cumulative maintenance performance, and cache freshness can be further improved by appropriately exploiting the temporal variations of data refreshing pattern and node contact process.

VIII. Conclusion

In this paper, we focus on maintaining the freshness of cached data in opportunistic mobile networks. Our basic idea is to let each caching node be only responsible for refreshing a specific set of caching nodes, so as to maintain cache freshness in a distributed and hierarchical manner. Based on the experimental investigation results on the refreshing patterns of real websites, we probabilistically replicate data updates, and analytically ensure that the freshness requirements of cached data are satisfied. The performance of our proposed scheme on maintaining cache freshness is evaluated by extensive trace-driven simulations on realistic mobile traces.

References

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN Routing As a Resource Allocation Problem. In *Proc. SIGCOMM*, 2007.

[2] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive wifi connectivity for moving vehicles. In *Proceedings of ACM SIGCOMM*, pages 427–438, 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] G. Cao. A scalable low-latency cache invalidation strategy for mobile environments. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1251–1265, 2003.

[6] J. Cao, Y. Zhang, G. Cao, and L. Xie. Data consistency for cooperative caching in mobile environments. *IEEE Computer*, 40(4):60–66, 2007.

[7] P. Cao and C. Liu. Maintaining strong cache consistency in the world wide web. *IEEE Transactions on Computers*, 47(4):445–457, 1998.

[8] G. Casella and R. Berger. *Statistical Inference*. Duxbury Press, 2001.

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

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

[11] V. Duvvuri, P. Shenoy, and R. Tewari. Adaptive leases: A strong consistency mechanism for the world wide web. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(4):1266–1276, 2003.

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

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

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-Tolerant Networking (WDTN)*, pages 244–251. ACM, 2005.

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

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

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

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

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

[26] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for http. *ACM SIGCOMM Computer Communication Review*, 27(4):181–194, 1997.

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

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

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

[30] A. Stuart, J. K. Ord, and S. Arnold. *Advanced Theory of Statistics: Classical Inference and the Linear Model*, volume 2A.

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

[32] 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.

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

[34] 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.