Social-Based Cooperative Caching in DTNs: A Contact Duration Aware Approach

Xuejun Zhuo*†, Qinghua Li†, Guohong Cao†, Yiqi Dai*, Boleslaw Szymanski‡, Tom La Porta†
*Tsinghua University, China, Email: zhuoxj07@mails.tsinghua.edu.cn, dyq@mail.tsinghua.edu.cn
†The Pennsylvania State University, USA, Email: {qxl118,gcao,tlp}@cse.psu.edu
‡Rensselaer Polytechnic Institute, USA, Email: szymab@rpi.edu

***Abstract*—Data access is an important issue in Delay Tolerant Networks (DTNs), and a common technique to improve the performance of data access is cooperative caching. However, due to the unpredictable node mobility in DTNs, traditional caching schemes cannot be directly applied. In this paper, we propose DAC, a novel caching protocol adaptive to the challenging environment of DTNs. Specifically, we exploit the social community structure to combat the unstable network topology in DTNs. We propose a new centrality metric to evaluate the caching capability of each node within a community, and solutions based on this metric are proposed to determine where to cache. More importantly, we consider the impact of the contact duration limitation on cooperative caching, which has been ignored by the existing works. We prove that the marginal caching benefit that a node can provide diminishes when more data is cached. We derive an adaptive caching bound for each mobile node according to its specific contact patterns with others, to limit the amount of data it caches. In this way, both the storage space and the contact opportunities are better utilized. To mitigate the coupon collector's problem, network coding techniques are used to further improve the caching efficiency. Extensive trace-driven simulations show that our cooperative caching protocol can significantly improve the performance of data access in DTNs.**

***Keywords*-Delay Tolerant Networks; Cooperative Caching; Contact Duration**

I. Introduction

In Delay Tolerant Networks (DTNs) [1], mobile nodes connect to each other using opportunistic contacts. Due to unpredictable node mobility, there is no end-to-end connection between mobile nodes, which greatly impairs the performance of data access.

Cooperative caching has been frequently used to improve the performance of data access in both wireline and wireless networks. However, due to the unstable topology and limited contact duration in DTNs, conventional cooperative caching techniques [2], [3], [4] may not be applicable to DTNs. For example, mobility is not well considered in the existing work, and the caching node may not be reachable by the data requester due to the unstable topology in DTNs. Further, existing studies all assume that as long as the data requester reaches the caching node (directly or through multi-hop), the complete data item can be retrieved. However, this assumption does not hold in DTNs, where the contact duration is usually short due to high node mobility. If too much data is cached at a node, it will be difficult for the node to send all the data to the requesters during the short contact period, thus wasting storage space. Therefore it is a challenge to determine *where to cache* and *how much to cache* in DTNs. We design a novel cooperative caching protocol to address these two issues.

To deal with the intrinsic unstable network topology in DTNs, we exploit the relatively stable social relations among nodes. Since nodes that belong to the same community have a high probability of meeting each other, it is more likely for a node to retrieve data from nodes with the same community than those outside. Among all the nodes within the same community, some may have higher potential to contribute their cached data to other community members. To increase the caching efficiency, a novel metric is proposed to measure the caching capability of the nodes and is used to determine where to cache the data.

Due to limited contact duration, a large data item may not be completely transmitted between two nodes during their contact. If the data is simply fragmented and only a part of it is transmitted during each contact, the well-known coupon collector's problem [5] will appear, which deteriorates the performance of the data access. To mitigate this problem, we adopt the random linear network coding technique [6] to encode the data. We then prove that the marginal caching benefit a node can provide diminishes when more data is cached. In other words, if too many packets are cached at a node, the caching efficiency decreases. Thus, we carefully consider how much data to cache at each node. To the best of our knowledge, this is the first paper to identify the effects of contact duration on cooperative caching in DTNs. More specifically, the contribution of the paper is three-fold:

1) To address the problem of *where to cache*, we exploit social relations among nodes, and propose a novel centrality metric to evaluate the caching capability of each node within the social community.
2) We identify the effects of contact duration on caching. To address the problem of *how much data to cache*, we derive an adaptive caching bound at each node based on its specific contact patterns with others.
3) We develop a distributed caching protocol, and demonstrate that it can significantly improve the performance of data access though trace-driven simulations.

This work was supported in part by Network Science CTA under grantW911NF-09-2-0053.

The rest of the paper is organized as follows. Section II introduces the network model. We then analyze the impact of contact duration on cooperative caching and present our contact Duration Aware Caching protocol (DAC) in Section III. Section IV evaluates our protocol via trace-driven simulations. Section V reviews the existing work, and Section VI concludes the paper.

II. Network Model

In this section, we describe the model we use to characterize the contact between mobile nodes, and give an introduction to our system.

A. Contact Graph

In DTNs, a relatively stable network contact graph $G(V, E)$ is usually used to describe the network. V is the set of nodes in the network, and E is the set of edges, with each edge $e_{ij} \in E$ representing the stochastic contact process between node pair $i, j \in V$. Similar to [7], [8], we consider that the pairwise inter-contact time, i.e., the time between two contacts, between nodes follows an exponential distribution. This modeling has also been experimentally validated in [8], [9]. The contacts between node i and node j can then be formulated as a Poisson process with the contact rate λ_{ij}, which represents the weight of the edge between the two nodes.

B. Contact Duration Limitation

Contact duration is a key factor in DTNs. This determines how much data can be exchanged during a contact. However, this factor has been overly simplified in most prior works. The assumptions on contact duration in DTNs, typically fall into two extremes: one assumes short contact duration where only one packet can be sent during a contact [10], and the other assumes long contact duration where an arbitrary number of packets can be sent during a contact [8], [11]. The protocols based on these overly simplified assumptions lead to either over-pessimistic or over-optimistic results, since some studies [12], [13] have experimentally validated that the contact duration in DTNs follows a Pareto distribution. In this paper, similar to [12], [13], we also assume that the contact duration between nodes follows Pareto distribution. Suppose the transmission rate is fixed, we define a random variable X_{ij} following the Pareto distribution to denote the amount of data that can be sent during a contact between node i and j.

C. Social Network Model

There are two key concepts in social network theory: *community* and *centrality*. Community is a coherent union; the nodes within the same community tend to have relatively regular and frequent contacts. Thus, we take advantage of such node behavior to perform cooperative caching, which enables data sharing within a community. Centrality measures the social power of a node based on how well it connects the network, and a node with higher centrality is more important to its community. There are three typical centrality metrics: 1) *'Degree Centrality'*, which represents the number of ties a node has to other nodes; 2. *'Closeness Centrality'*, which represents the distance between a node and other nearby nodes; 3. *'Betweenness Centrality'*, which represents the extent to which a node lies on the shortest paths linking to other nodes. These metrics do not directly represent the probability that nodes contact others and the contact duration limitations, both of which reflect the potential of the node to contribute its cached data to the community. Thus, we provide a new centrality metric to evaluate the caching capability of the nodes.

D. System Introduction

In this paper, we consider a hybrid network scenario which includes Access Points (APs) and mobile nodes. The APs providing data downloading service are located in the Internet infrastructure, which only cover a small fraction of the network area. The connections between mobile nodes and APs are intermittent and opportunistic. A mobile node can directly download data when it has a connection with an AP. When the node has no direct connection with any AP, it can only download data via a multi-hop relay provided by other mobile nodes[1]. We aim to design a cooperative caching protocol in such a scenario, to let some nodes cache the data and act as the source for future requests. In this way, the performance of data access can be significantly improved.

Due to the contact duration limits, a complete data item may not be transmitted during a contact. If the data item is simply fragmented into multiple consecutive native packets, to recover the original data, a node has to collect all the native packets. This suffers from the well-known coupon collector's problem [5]. Specifically, suppose there are s distinct native packets and each can be collected with an equal probability. Then it has been proven that the expected number of packets a requester needs to collect to accumulate all s distinct packets is on the order of $\Theta(s \log s)$.

To mitigate the problem caused by the simple fragmentation, we adopt the random linear network coding [6]. At first, the AP cuts the data item P into s uniformly sized native packets, $P = \{p_1, p_2, \cdots, p_s\}$, and randomly generates a set of coding coefficient vectors $\vec{\alpha_j} = (\alpha_{j1}, \alpha_{j2}, \cdots, \alpha_{js})$ from a Galois Field to compute the random linear combination of the native packets: $p'_j = \sum_{i=1}^{s} \alpha_{ji} p_i$. Note that all generated coded packets are also uniformly sized. Then the AP sends these coded packets to the data requester via direct contact or by multi-hop relay. Unlike erasure coding which only allows the data source to encode packets, random linear network coding also allows intermediate nodes to perform encoding.

[1]Any DTN routing protocol can be used in our scenario for end-to-end communication, and it is not the focus of this paper.

Thus it can greatly reduce the possibility that different nodes cache duplicate packets. Specifically, if an intermediate node receives s' coded packets, it also randomly generates a set of coding coefficient vectors $\vec{\beta_k} = (\beta_{k1}, \beta_{k2}, \cdots, \beta_{ks'})$ and computes the random linear combination of the received packets $p''_k = \sum_{j=1}^{s'} \beta_{kj} p'_j = \sum_{j=1}^{s'} \beta_{kj} \sum_{i=1}^{s} \alpha_{ji} p_i = \sum_i (\sum_j \beta_{kj} \alpha_{ji}) p_i$. The new random linear combination of coded packets is also a random linear combination of the native packets. As long as the requester collects any s linearly independent coded packets, the original data item can be recovered by using matrix inversion. In this way, the efficiency of data access is greatly improved.

III. Caching Protocol Design

In this section, we analyze the effects of contact duration on cooperative caching, and propose a distributed contact Duration Aware Caching (DAC) protocol from a social network perspective.

Since the nodes in the same community have a higher probability to share their cached data, we exploit this social property to determine the caching solution within each community. Also, it is impractical to maintain the global knowledge of the network at every node, but community-based caching only depends on each node's local knowledge about the community. For simplicity, we assume that the requester can be any node in the community.

A. Main Idea

To increase the caching efficiency, the nodes with larger potential to contribute the cached data to others deserve higher priority to be selected as the caching node. When evaluating the potential, traditional caching solutions all assume that the data requester can always get the complete requested data item from the caching node when a connection arises between them. Thus, these solutions take the complete data item as the caching unit; i.e., a node either caches the complete data item or does not cache any of it. However, this assumption does not hold in DTN scenarios. Due to the contact duration limits in DTNs, the amount of data that can be transmitted during a contact is restricted. Under this condition, the caching efficiency is reduced if a node caches too many packets, since the limited contact duration prohibits it from transmitting all of them.

Fig. 1 shows an example to illustrate the impact of the contact duration limits on caching. Node A is requesting a data item with eight packets, and it will sequentially contact nodes B, C, and D when it moves. In Fig. 1(a), the complete data item is taken as the caching unit. Node B has been selected as the caching node. Under this caching solution, node A can only retrieve five packets from B due to the contact duration limitation. As a result, the remaining three packets cached in node B have not been utilized, and A's contact opportunities with node C and D have been wasted. However, when packet is taken as the caching unit and we can let node B, C, and D to cache five, two, and one packet, as shown in Fig. 1(b), node A can retrieve all eight packets. Under this caching solution both the storage buffer and the contact opportunities are better utilized. As can be seen, in the contact duration limited scenarios, rather than taking the complete data as the caching unit, the problem of how many packets to cache at each node should be carefully determined, to improve the caching efficiency.

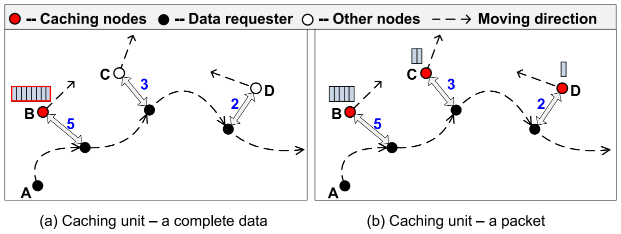


Figure 1. The impact of contact duration limitation on caching. The number marked next to the two-way arrows denotes the number of packets that can be transmitted during the contact.

In the rest of the paper, we refer the caching solution which only solves the problem of where to cache and takes the complete data as the caching unit as the *naive scheme*. Compared to it, our goal is to solve both the problems of where to cache and how many packets to cache at each node, based on the following three intuitions:

1) A centrality metric should be defined to evaluate the marginal caching benefit that a node can provide to its community. The node with higher centrality deserves larger chance to be selected as a caching node.
2) The short contact duration limits the caching benefit that a caching node can provide to its community, and a novel caching bound should be set for each caching node according to its contact pattern.
3) The number of caching replicas of a data item within a community should be limited. In this way, the caching scheme is prevented from taking up too much of the users's storage space so that it may be shared by other applications.

B. Problem Formulation and Analysis

In DTNs, the data downloading delay is much longer compared to the traditional networks, but it is still impractical for a requester to download data without any time limit. Thus, we set a time constraint T for the data downloading. To measure the caching efficiency, we define the *caching benefit* provided by a caching node as the expected amount of data that it can send to the requester before the time constraint. For example, if node i can send s_i coded packets to node j with probability p_{ij}, before the time constraint, the caching benefit provided by node i to node j is $B_{ij}(s_i) = g s_i p_{ij}$, where g is the size of one coded packet. We use the term *social caching benefit* to represent the total caching benefits provided by all the caching nodes to a randomly chosen node in the community, which can be calculated as:

$$\sum_{i=1}^{N} \overline{B_i}(s_i) = \sum_{i=1}^{N} \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} B_{ij}(s_i) \tag{1}$$

where N is the total number of nodes in the community, and $\overline{B_i}(s_i)$ denotes the caching benefit provided by node i to a randomly chosen node in the community. The larger the social caching benefit is, the fewer packets need to be downloaded by a random requester from the APs, which results in a shorter delay for it to reconstruct the complete original data, and thus an increase of the successful data delivery ratio before the time constraint.

We further define the *caching cost* r as the number of caching replicas in the network, which is the ratio of the total number of coded packets cached in the community to the number of native packets that the data contains. For example, if a data contains 4 native packets and there are 10 coded packets of this data cached in the community, the caching cost is $r = 10/4 = 2.5$. We aim to determine the optimal caching solution s_i for $i = 1, \cdots, N$, where s_i represents the number of coded packets cached at node i. Let s_{data} denote the number of native packets that the data contains. The formal definition of the caching problem is given as:

Definition 1. *The cooperative caching problem is to determine the caching solution s_i for $i = 1, \cdots, N$, which maximizes the social caching benefit, subject to a fixed upper bound of the caching cost within a community.*

$$\max \quad \sum_{i=1}^{N} \overline{B_i}(s_i) \tag{2}$$

$$s.t. \quad \sum_{i=1}^{N} s_i \leq r s_{data} \tag{3}$$

$$\forall i \;\; s_i \leq s_{data}. \tag{4}$$

Before analyzing the problem in the contact duration limited scenario, we first study it in a toy scenario without a contact duration limitation for comparison purpose. To evaluate the caching capability of a node, we define a novel *centrality* metric to measure the marginal caching benefit it can provide to its community. In economics, the *marginal utility* of a product is the extra utility gained from consuming an additional unit of that product. In this paper, we borrow this concept, and define the marginal caching benefit as the gain in the caching benefit provided by the node as a result of caching an additional packet. We will prove that without considering the contact duration limitation, the centrality of a node is purely dependent on its mobility pattern. As a result, the active nodes should cache as much data as possible. However, when the contact duration is limited, the node's centrality decreases with the increase of the total number of packets it has already cached.

1) Toy Scenario (Without Contact Duration Limitation): Without a contact duration limitation, an arbitrary number of packets can be transmitted when two nodes contact. We use a random variable U_{ij} to represent the total amount of data transmitted from a caching node i to node j before time constraint T. As mentioned in Section II-A, the contact between a node pair can be modeled as a Poisson process, and thus U_{ij} can be calculated as:

$$U_{ij} = \begin{cases} g s_i & w.p. \;\; 1 - e^{-\lambda_{ij} T} \\ 0 & w.p. \;\; e^{-\lambda_{ij} T} \end{cases} \tag{5}$$

where g is the size of one coded packet. Thus the caching benefit that node i can provide to a randomly chosen node in the community can be calculated as:

$$\begin{aligned} \overline{B_i^{toy}}(s_i) &= \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} B_{ij}^{toy}(s_i) \\ &= \frac{g s_i}{N-1} \sum_{j=1, j\neq i}^{N} (1 - e^{-\lambda_{ij} T}) \end{aligned} \tag{6}$$

where $B_{ij}^{toy}(s_i) = E(U_{ij})$. To evaluate the caching capability of each node, we define a centrality metric which represents the marginal caching benefit provided by the node. Note that the marginal benefit is normalized by g, the size of one coded packet.

Definition 2. *In the toy scenario, the centrality of node i which has already cached s_i coded packets is defined as:*

$$\begin{aligned} CEN_i^{toy}(s_i) &= \frac{\Delta \overline{B_i^{toy}(s_i)}}{g \Delta s_i} \\ &= \frac{\overline{B_i^{toy}}(s_i + 1) - \overline{B_i^{toy}}(s_i)}{g} \\ &= \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} 1 - e^{-\lambda_{ij} T}. \end{aligned} \tag{7}$$

As can be seen, in the toy scenario the centrality of a node is independent of s_i, and is only determined by the node's mobility pattern. Based on the centrality metric, a cached packet always produces a higher benefit if it is cached in a node with higher centrality. Thus, the optimal solution of the problem defined in Definition 1 can be derived via a greedy algorithm; i.e., we select the node with the maximum centrality round by round to cache the complete data, until the caching cost is reached. Note that the naive scheme achieves the optimal solution in the toy scenario.

2) Realistic Scenario (With Contact Duration Limitation): In real DTN scenarios, the contact duration limitation cannot be ignored. According to Section II-B, we define a random variable X_{ij} which follows Pareto distribution to represent the amount of data sent within one contact between node i and j. Let a random variable Y_{ij} which follows Poisson distribution denote the number of contacts between the two nodes before time constraint T. Thus, $X_{ij}^{(1)}, X_{ij}^{(2)}, \cdots, X_{ij}^{(Y_{ij})}$ are Y_{ij} i.i.d. variables which represent the amount of data sent in each contact between them. Then, the total amount of data U_{ij} sent from node i to j before time constraint T can be calculated as follows:

$$U'_{ij} = X_{ij}^{(1)} + X_{ij}^{(2)} + \cdots + X_{ij}^{(Y_{ij})}$$
$$U_{ij} = \begin{cases} U'_{ij} & U'_{ij} \le gs_i \\ gs_i & U'_{ij} > gs_i \end{cases} \tag{8}$$

where the second condition specifies that the amount of data sent from node i to j cannot exceed the total amount of data cached at node i. Thus, the caching benefit that node i provides to node j can be calculated as:

$$\begin{aligned} B_{ij}^{real}(s_i) &= E(U_{ij}) \\ &= \int_0^{gs_i} u f_{U'_{ij}}(u)\mathrm{d}u + \int_{gs_i}^{\infty} gs_i f_{U'_{ij}}(u)\mathrm{d}u \end{aligned}$$

where $f_{U'_{ij}}(u)$ is the Probability Density Function (PDF) of the random variable U'_{ij}. Recall in the previous toy scenario, the benefit $\overline{B_i^{toy}}(s_i)$ is linearly proportional to s_i, given by Eqn. 6. However, this is not the case for $\overline{B_i^{real}}(s_i)$ in the realistic scenario with the contact duration limitation.

Lemma 1. $B_{ij}^{real}(s_i)$ *is a monotonically increasing concave function of* s_i.

Proof: This lemma is proved in the Appendix. ■

The caching benefit that node i provides to a randomly chosen node in the community can be represented as: $\overline{B_i^{real}}(s_i) = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N} B_{ij}^{real}(s_i)$. According to lemma 1, we derive that $\frac{d(\overline{B_i^{real}}(s_i))}{d(s_i)} > 0, \frac{d^2(\overline{B_i^{real}}(s_i))}{d(s_i^2)} < 0$. Therefore, the marginal caching benefit provided by node i, $\frac{\Delta\overline{B_i^{real}}(s_i)}{g\Delta s_i}$, decreases with the increase of s_i.

Unfortunately, it is pretty involving to derive a closed form expression of $B_{ij}^{real}(s_i)$, since U'_{ij} is the summation of a random number of random variables. Next, we give an approximated closed form expression to $\overline{B_i^{real}}(s_i)$ instead.

As we have proved, in the realistic scenario with contact duration limitation, the caching benefit that node i can provide is a monotonically increasing concave function of s_i as shown in Figure 2. The curve of $\overline{B_i^{real}}(s_i)$ is below the curve of $\overline{B_i^{toy}}(s_i)$, since the short contact duration in the real scenario limits the caching benefit. Besides, $E(U'_i)$ is the upper bound of $\overline{B_i^{real}}(s_i)$, since it represents the expected amount of data transmitted from node i to a randomly chosen node in the community when s_i is infinite. When the power parameter of X_{ij} is larger than 1, we can derive the closed form expression of $E(U'_i)$ as $E(U'_i) = \frac{\sum_{j=1,j\neq i}^{N} E(U'_{ij})}{N-1} = \frac{\sum_{j=1,j\neq i}^{N} \lambda_{ij} T \frac{x_{ij}\tau_{ij}}{\tau_{ij}-1}}{N-1}$, where x_{ij} is the scale parameter of X_{ij}, which represents its minimum possible value, and τ_{ij} is the power parameter of X_{ij}, which defines the shape of distribution.

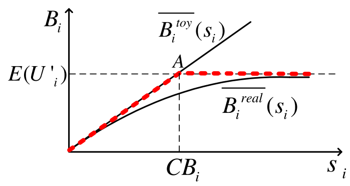


Figure 2. Caching benefit of node i as a function of the caching size s_i

We use the red dotted curve as drawn in Figure 2 to approximate $\overline{B_i^{real}}(s_i)$, and rename it as $\overline{B_i'^{real}}(s_i)$. We can see that $\overline{B_i'^{real}}(s_i)$ is proportionally increasing with the increase of s_i before reaches $E(U'_i)$, and remains at $E(U'_i)$ when s_i further increases. We define the projection of the critical point A of $\overline{B_i'^{real}}(s_i)$ (the red dotted curve) on axis s_i as the Caching Bound (CB) of node i, which can be calculated as:

$$CB_i = \frac{\sum_{j=1,j\neq i}^{N} E(U'_{ij})}{g\sum_{j=1,j\neq i}^{N}(1-e^{-\lambda_{ij}T})}. \tag{9}$$

Thus, the caching benefit of node i can be re-represented as the following piecewise function:

$$\overline{B_i'^{real}}(s_i) = \begin{cases} gs_i \frac{\sum_{j=1,j\neq i}^{N}(1-e^{-\lambda_{ij}T})}{N-1} & s_i < CB_i \\ E(U'_i) & s_i \ge CB_i. \end{cases}$$

Based on the above analysis, the centrality metric can be re-defined as:

Definition 3. *In the realistic scenario, the centrality of node* i *which has already cached* s_i *coded packets is defined as:*

$$\begin{aligned} CEN_i^{real}(s_i) &= \frac{\Delta\overline{B_i^{real}}(s_i)}{g\Delta s_i} \\ &= \frac{\overline{B_i^{real}}(s_i+1) - \overline{B_i^{real}}(s_i)}{g} \\ &= \begin{cases} \frac{\sum_{j=1,j\neq i}^{N}(1-e^{-\lambda_{ij}T})}{N-1} & s_i < CB_i \\ 0 & s_i \ge CB_i. \end{cases} \end{aligned}$$

The centrality of a node in the realistic scenario has the same form as that in the toy scenario, when s_i has not reached the node's caching bound. When the caching bound is reached, the centrality value turns to 0 which means that no caching benefit gain can be brought by caching more data at the node.

Before introducing our caching protocol in realistic scenario, we describe the optimal solution of the problem defined in Definition 1. The optimal solution can also be obtained through a greedy algorithm. Specifically, we first sort the nodes in the decreasing order of their centrality. Then we iteratively select the node with the maximal centrality to cache the data until its caching bound is reached, after which its centrality turns to 0. Such iteration stops until the total caching cost within a community is reached.

C. The Contact Duration Aware Caching Protocol (DAC)

In this section, we present our distributed contact Duration Aware Caching protocol (DAC). To begin with, we first illustrate how to detect social community in DTNs.

1) Community Detection: We adopt the distributed *k-clique* algorithm proposed by Hui *et al.* [14] to identify the communities. The main idea of the algorithm is that each node maintains a *familiar set* containing its contacted neighbors, based on a specific criteria, and then builds its local *community set* by merging its familiar set with other selected nodes which share at least $k-1$ common 'neighbors'. In this paper, we use the contact rate as the criterion for familiar set detection. In other words, only if the contact rate between two nodes is higher than certain threshold, they belong to each other's familiar set.

2) Protocol Design: Each node maintains a Community Information Table (CIT) which stores the local knowledge of the node ID, centrality, caching bound, and caching inventory of other nodes in the same community. The caching inventory contains the $rank$; i.e., the number of linearly independent coded packets, also known as the *innovative packets*, of the data item cached at the corresponding node. Each record in CIT is time-stamped to show its freshness. The format of CIT is shown in Table I.

When two nodes belong to the same community contact, they exchange their CITs as well as their own time-stamped local information including their centrality, caching bound and caching inventory. Based on the received metadata, each node updates its own CIT. Specifically, it first updates the corresponding record of the contacted node maintained in its own CIT, and then merges the received CIT with its own CIT. During the merge, the records with an older timestamp are replaced by the ones with a newer timestamp.

Table I
COMMUNITY INFORMATION TABLE

Node ID	Centrality	Caching Bound	Inventory	Timestamp
j	CEN_j	CB_j	$rank_j$	t

In DAC, a node has two ways to become a caching node: 1) When the node acts as a relay or a requester, a passive caching may be performed at the node; 2) When the node contacts a caching node with lower centrality, an active caching reallocation may be performed between them to increase caching efficiency. Next, we illustrate the details.

When a requester or a relay receives some packets, it passively decides whether to cache them locally. If the total caching cost of the data within the community is reached, the node will not cache them. Otherwise, it checks the dependency of the received packets one by one, and only caches the innovative packets with a probability proportional to its centrality value. The node caches the innovative packets until its caching bound is reached. Once this happens, the node encodes the remaining innovative packet p'_x with each of its locally cached packet p'_k as: $p'_k = p'_k + \beta_k p'_x$, where β_k is a coding coefficient randomly generated from a Galois Field. In this way, the node's cached packets are further diversified to have higher chance of being innovative to the coded packets cached at others.

Due to the unstable network topology in DTNs, those nodes that passively cache the data may have relatively low centrality. When a caching node with low centrality contacts another node which belongs to the same community and has higher centrality, the former transfers the cached data to the latter to increase the caching efficiency, namely by performing an active caching data reallocation. More specifically, when a caching node i contacts node j in the same community, the two nodes first exchange the metadata. Note that since the size of the metadata is much smaller than the real data size, so we ignore its cost. Then, node i updates its CIT based on the received information, and re-performs the greedy algorithm described in Section III-B2, if the local information changes. There are four possible results of the caching node selection based on node i's local view:

1) Both node i and node j are selected;
2) Node i is not selected, but node j is;
3) Neither node i nor node j is selected;
4) Node i is selected, but node j is not.

In case 1), node i randomly generates a set of linearly independent coded packets by encoding all its cached coded packets via random linear network coding, and sends them to nodes j until the contact ends. In case 2), node i does the same thing as in case 1), but after sending a set of coded packets to node j, node i deletes the same number of the coded packets from its local caching buffer. Let s_{del} denote the number of coded packets that node i needs to delete. To diversify the cached packets, node i does not randomly delete some packets but encodes all the packets cached in its buffer to generate $s_i - s_{del}$ new coded packets and locally cache them. In case 3), node i checks the caching cost in the community. If the caching cost is reached, it deletes all the packets cached in its buffer. Finally, in case 4) node i does not do anything.

When node j receives the packets from node i, it caches the innovative packets. If node j's caching bound is reached, it encodes the received packet to every locally cached packet.

Similarly, such caching data reallocation process is also executed from node j to i when two nodes contact, if node j is a caching node.

IV. PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of DAC through trace-driven simulations.

A. Simulation Settings

Our performance evaluations are performed on both the *MIT Reality* trace which is gathered by the MIT Reality Mining Project [15] and the *Infocom2006* trace which is collected by the Haggle Project [16]. These traces record contact history of users who carry Bluetooth devices. The Bluetooth devices periodically detect their neighbors and record the contact information, including two contact parties, the start time, and the duration. The detailed characteristics of the two traces are summarized in Table II.

Table II
SUMMARY OF FOUR REALISTIC DTN TRACES

	Infocom06	MIT
Device	iMote	Cellphone
Trace duration(days)	4	246
Granularity(sec)	120	300
# of devices	78	97
# of internal contacts	146193	54667
Avg. # of contacts/pair/day	6.7	0.024

In both traces, we generate a virtual static node to play the role of the AP. The contact processes between the AP and the mobile nodes follows Poisson distribution, and the contact rate is generated within the range of $[1.2\times10^{-7}, 1.2\times10^{-6}]$ for the MIT Reality trace and $[3.6\times10^{-5}, 1.2\times10^{-4}]$ for the Infocom2006 trace. We let the majority of the mobile nodes have low contact rates with the AP, and only a small portion of mobile nodes have high contact rates. The contact duration between the mobile nodes and the AP follows Pareto distribution. The parameters of the distribution are set the same as that of the aggregated contact duration distribution between the nodes derived from the traces. We assume that every node has enough contact history to estimate its contact duration distribution and contact rate with others. Due to the coarse granularity of the two traces, there exist many contact records whose contact duration is zero. In the simulations, we set the granularity of the trace as the time unit of transmitting one packet, and if the contact duration is zero, we assume that only one packet can be transmitted. At each 50 time units, a random node is selected as data requester.

In the simulations, we generate the random coefficients for each linear combination from a Galois Field $GF(2^8)$, and set the batch size to 32, i.e., as long as a requester collects 32 linearly independent packets, it can recover the original data. In each simulation run, the first $1/3$ of the MIT Reality trace and the first $1/4$ of the Infocom2006 trace are used for warm-up. To avoid end-effects, no data request will be generated in the last $1/6$ of each trace. The presented results are averaged over 10 runs with different random seeds.

B. Routing Protocol and Evaluation Metrics

We adopt a typical benchmark algorithm PROPHET [17] as the routing protocol for the AP to send data to the requester, and use the same parameters as in [17]. PROPHET calculates a delivery predictability metric based on the contact histories, and then relays the packet to the nodes with a higher metric.

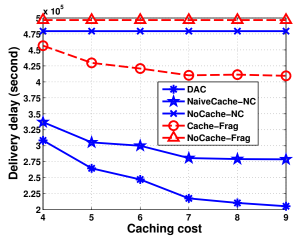
To evaluate the performance of the caching protocol, we use the following two metrics: 1) **Delivery delay:** the average delay for a requester to receive the complete data item before the time constraint; 2) **Delivery ratio:** the ratio of the number of data requesters receiving the complete data item before the time constraint to the total number of requesters.

C. Schemes for Comparison

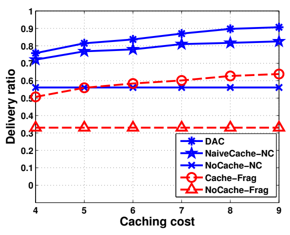
To better understand the performance of our protocol, we compare our protocol to the following protocols which isolate the impact of various features, such as the adaptive caching bound and the network coding: 1) **NoCache-Frag**: No cooperative caching (requesters download from the AP using PROPHET), with simple fragmentation (i.e., without network coding). 2) **Cache-Frag**: Cooperative caching, with simple fragmentation. 3) **NoCache-NC**: No cooperative caching, with network coding. 4) **NaiveCache-NC**: Cooperative caching, no caching bound for nodes, with network coding. Our scheme with all the features is denoted as **DAC**.

D. Evaluation Results

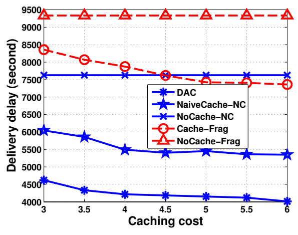
Compared to the MIT Reality trace, the Infocom2006 trace has higher network density and contact rate. In the Infocom2006 trace, people all attend the same event, and thus they tend to fall into the same community. We set the threshold for the community detection to 9×10^{-5}. However, in the MIT Reality trace, the scale of community varies obviously under different community detection thresholds. We set the threshold to 2.5×10^{-7} in the trace, which is low enough to maintain stable communities, and we choose the biggest community to show the caching performance.



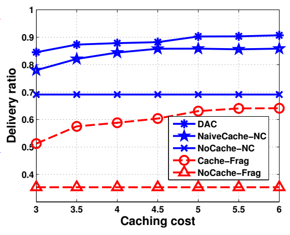
(a) Delivery delay - *MIT Reality*



(b) Delivery ratio - *MIT Reality*



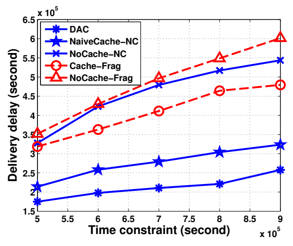
(c) Delivery delay - *Infocom06*



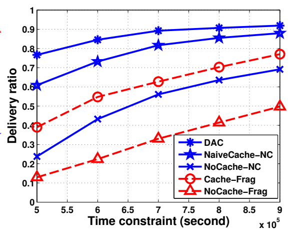
(d) Delivery ratio - *Infocom06*

Figure 3. Caching performance as a function of the caching cost

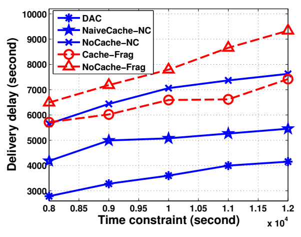
1) The Effects of Caching Cost: Figure 3 shows the delivery delay and the delivery ratio of the five schemes as a function of the caching cost, i.e., the number of replicas that the data item can have. The caching cost is changed from 4 (which means a total of $4 \times 32 = 128$ coded packets can be cached) to 9 in the MIT Reality trace, and from 3 to 6 in the Infocom2006 trace. Obviously, for the three schemes with cooperative caching: DAC, NaiveCache-NC, and Cache-Frag, when the caching cost increases, the delivery delay decreases and the delivery ratio increases accordingly. As can be seen, such change is more obvious when the caching cost is relatively small. When the caching cost further increases (to 8 in the MIT reality trace, and 5 in the Infocom2006 trace), the limited contact opportunities become the performance bottleneck, and then the delivery delay and the delivery ratio have no obvious improvement even with more caching copies. Thus, in the following simulations, we set the caching cost in the MIT Reality and the Infocom2006 traces to 8 and 5 respectively.



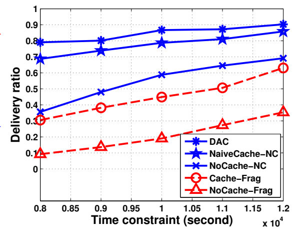
(a) Delivery delay - *MIT Reality*



(b) Delivery ratio - *MIT Reality*



(c) Delivery delay - *Infocom06*



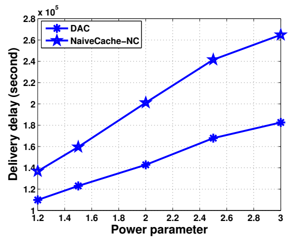
(d) Delivery ratio - *Infocom06*

Figure 4. Caching performance as a function of the time constraint

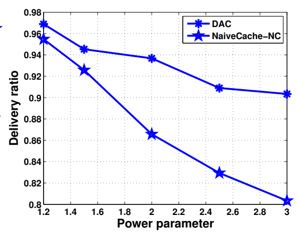
2) The Effects of Time Constraint: Next, we evaluate the caching performance as a function of the time constraint. Since the Infocom2006 trace has much higher contact rate, we set a shorter time constraint compared to the MIT Reality trace. More specifically, we change the time constraint from 5×10^5 to 9×10^5 seconds in the MIT Reality trace, and from 0.8×10^4 to 1.2×10^4 seconds in the Infocom06 trace.

As can be seen from Figure 4, the schemes with caching outperform the schemes without caching, and the schemes using network coding outperform the schemes using simple fragmentation. Further, DAC achieves the lowest delivery delay and the highest delivery ratio among all the five schemes. Compared to the other four schemes, NaiveCache-NC, NoCache-NC, Cache-Frag, NoCache-Frag, DAC reduces the average delay in the MIT Realiry trace (Infocom2006 trace) by about 23% (29%), 54% (48%), 48% (45%), 56% (55%), and improves the average delivery ratio by about 12% (10%), 91% (60%), 48% (96%), 229% (395%).

Compared to the NaiveCache-NC scheme that uses cooperative caching and network coding but treats a complete data as the caching unit without setting a caching bound, our scheme performs better with the same caching cost. This is because in NaiveCache-NC, too many packets are cached at the nodes with high centrality, but due to the limited contact duration, it is hard for them to send all the cached data to the requesters. Thus, the caching efficiency decreases and some storage space is wasted. Also, given a fixed caching cost, the data is cached at a few hop-spot nodes in NaiveCache-NC. Thus the probability for the data requester to contact those nodes is smaller, and some contact opportunities are wasted. In contrast, our caching scheme sets an adaptive caching bound to every node according to its contact patterns, and thus more nodes with higher centrality are selected as the caching nodes. In this way, the limited contact opportunities and storage space are better utilized, and the caching efficiency is improved.



(a) Delivery delay



(b) Delivery ratio

Figure 5. Caching performance as a function of the power parameter

3) The Effects of Contact Duration: The course granularity of the MIT Reality trace has introduced some inaccuracies to contact duration detection, especially for short contacts. Also, we expect to see the impact of different contact durations on our protocol. Thus, in this subsection, we generate synthetic contact durations for the MIT Reality trace. The synthetic contact duration between node pairs follows Pareto distributions, and we change the power parameter of the distribution to adjust the expected length of the contact duration. According to the Pareto distribution, when the power parameter increases, the expected contact duration will decrease.

Figure 5 shows the performance of DAC and NaiveCache-NC when the power parameter changes from 1.2 to 3.0. As can be seen, when the power parameter is small, i.e., the contact duration constraint is not so strict, the performance gain provided by our scheme compared to NaiveCache-NC scheme is relatively small. For example, when the power parameter is 1.2, our scheme reduces the delay by 20%, and improves the delivery ratio by 1.5% compared to NaiveCache-NC. This is because in the scenario with relatively long contact duration, more data can be transmitted during a contact, and then the caching bounds calculated in our scheme tend to be large. Thus, the inefficiency of NaiveCache-NC without considering the contact duration limitation is not so obvious. However, when the power parameter increases, the short contact duration results in smaller caching bounds, and the advantage of our scheme over the NaiveCache-NC becomes more obvious. For example, when the power parameter reaches 3, our scheme reduces the delay by 31% and improves the delivery ratio by 12% compared to NaiveCache-NC. In summary, our caching protocol is adaptive to the variation of the contact duration, and the stricter the contact duration limitation is, the larger the improvement that our caching protocol provides.

V. Related Work

Data forwarding in DTNs has attracted a lot of attention. Recent studies improve the performance of data forwarding by exploiting the advanced node behaviors. In [18], a special kind of node with desirable mobility patterns is added to the network to act as message ferries for data delivery. In [19], [20], the authors defined social based metrics for relay node selection. In [21], the authors proposed protocols to improve the performance of data forwarding by exploiting the transient social contact patterns. They formulated the transient social contact patterns based on experimental studies of realistic DTN traces, and proposed appropriate forwarding metrics based on these patterns to improve the effectiveness of data forwarding.

Data access is another important topic in DTNs. Existing work in this area focuses on the push-based model: the source actively pushes the data to as many interested nodes as possible. In [22], a theoretical analysis on data dissemination in DTNs has been conducted. In [8], the authors studied multicast in DTNs from a social network perspective, where the source disseminates the data to multiple known destinations. The studies in [11], [23] are based on a pub-sub structure, in which the destination of each data item is determined based on the nodes' subscriptions to the channels. All these works treat data recipients as a priori knowledge, and push data to them. However, in many scenarios, it is more beneficial for the interested nodes to actively download the necessary data. In order to reduce the downloading delay, cooperative caching is frequently required. In [2], Yin and Cao proposed a cooperative caching protocol in MANETs, which caches the data or the data path at certain relay nodes to reduce the delay of future requests. In [24], [25], the authors reduced the data redundancy among neighboring nodes to create content diversity with the limited cache space. However, these existing caching schemes are not applicable to DTNs due to the intrinsic challenging characteristics of DTNs. In [26], Gao *et al.* proposed to intentionally cache data at a set of network central locations which can be easily accessed by other nodes in the network. However, their cooperative cache solution did not consider the contact duration limitation.

To deal with the limited contact duration in DTNs, one solution is to cut the data into a bunch of smaller native packets. For example, Pitkanen *et al.* designed proactive and reactive fragmentation schemes to support partial message transfer in DTNs [27]. In order to mitigate the coupon collector's problem brought by fragmentation, in [28], the authors adopted erasure coding to encode the original data to a set of smaller coded packets, and split the coded packets to multiple paths. Also, Lee *et al.* adopted network coding to design a file swarming protocol for VANETs [29]. Lin *et al.* provided a stochastic analysis framework to study the performance of epidemic routing when random linear network coding and pure fragmentation are used [30].

VI. Conclusions

In this paper, we identified the effects of the contact duration limitation on cooperative caching in DTNs. Our theoretical analysis shows that the marginal caching benefit that a caching node can provide diminishes when it caches more data. Based on this observation, we have designed a contact Duration Aware Caching (DAC) protocol, which exploits social network concepts to address the challenge of the unstable network topology in DTNs. Trace-driven simulations show that by adopting DAC, the performance of data access can be significantly improved.

To the best of our knowledge, this is the first paper to identify the effects of contact duration on caching in DTNs. As the initial work, we do not expect to solve all the problems. In this paper, we have addressed the problem of where to cache and how much data to cache, given a fixed number of replicas for a data item. As future work, we will consider how to determine the optimal number of replicas for each data item. This problem is intuitively related to the following four factors: 1) data access pattern, 2) data size, 3) node mobility pattern, and 4) storage limitation. Also, we will deal with new challenges of considering multiple data items: 1) How to determine the priority of the data items when they compete for the limited contact duration and caching space? Intuitively, the priority of a data item is determined by its popularity and redundancy in the community. The popular but scarce data items deserve high priority. 2) Where to cache when the storage space of the nodes with high centrality is full? One simple solution is to let the high centrality nodes transfer some data items with low priority to the node with lower centrality to make room for the data items with high priority.

References

[1] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. of ACM SIGCOMM*, 2003.

[2] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 1, pp. 77–89, 2006.

[3] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Caching in Wireless P2P Networks: Design, Implementation, and Evaluation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 229–241, 2010.

[4] B. Tang, H. Gupta, and S. R. Das, "Benefit-based Data Caching in Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 3, pp. 289–304, 2008.

[5] M. Mitzenmacher and E. Upfal, "Probability and Computing: Randomized Algorithms and Probabilistic Analysis," *Cambridge*, 2008.

[6] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. of Annual Allerton Conf. on Comm., Control, and Comput.*, 2003.

[7] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resouce Allocation Problem," in *Proc. of ACM SIGCOMM*, 2007.

[8] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in Delay Tolerant Networks: A Social Network Perspective," in *Proc. of ACM MOBIHOC*, 2009.

[9] V. Conan, J. Leguay, and T. Friedman, "Characterizing Pairwise Inter-Contact Patterns in Delay Tolerant Networks," in *Proc. of Autonomic Computing and Communication Systems*, 2007.

[10] Y. Lin, B. Li, and B. Liang, "Efficient Network Coded Data Transmissions in Disruption Tolerant Networks," in *Proc. of IEEE INFOCOM*, 2008.

[11] C. Boldrini, M. Conti, and A. Passarella, "ContentPlace: Social-aware Data Dissemination in Opportunistic Networks," in *Proc. of ACM MSWiM*, 2008.

[12] W. Wang, V. Srinivasan, and M. Motani, "Adaptive Contact Probing Mechanisms for Delay Tolerant Applications," in *Proc. of ACM MOBICOM*, 2007.

[13] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Pocket Switched Networks: Realworld mobility and its consequences for opportunistic forwarding," Tech. Rep. UCAM-CL-TR-617, 2005.

[14] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed Community Detection in Delay Tolerant Networks," in *Proc. of ACM MOBIARCH*, 2007.

[15] N. Eagle and A. Pentland, "Reality Mining: Sensing Complex Social Systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, 2006.

[16] "Haggle Project." [Online]. Available: http://www.haggleproject.org

[17] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE CCR*, vol. 7, no. 3, pp. 19–20, 2003.

[18] W. Zhao, M. Ammar, and E. Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," in *Proc. of ACM MOBIHOC*, 2004.

[19] E. Bulut and B. K. Szymanski, "Friendship Based Routing in Delay Tolerant Mobile Social Networks," in *Proc. of IEEE GLOBECOM*, 2010.

[20] E. Daly and M. Haahr, "Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs," in *Proc. of ACM MOBIHOC*, 2007.

[21] W. Gao and G. Cao, "On Exploiting Transient Contact Patterns for Data Forwarding in Delay Tolerant Networks," in *Proc. of IEEE ICNP*, 2010.

[22] C. Boldrini, M. Conti, and A. Passarella, "Modelling Data Dissemination in Opportunistic Networks," in *Proc. of ACM CHANTS*, 2008.

[23] F. Li and J. Wu, "Mops: Providing Content-based Service in Disruption Tolerant Networks," in *Proc. of IEEE ICDSC*, 2009.

[24] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," in *Proc. of INFOCOM*, 2001.

[25] M. Fiore, F. Mininni, C. Casetti, and C. Chiasserini, "To Cache or Not To Cache?" in *Proc. of IEEE INFOCOM*, 2009.

[26] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Supporting Cooperative Caching in Disruption Tolerent Networks," in *Proc. of IEEE ICDCS*, 2011.

[27] M. Pitkanen, A. Keranen, and J. Ott, "Message Fragmentation in Opportunistic DTNs," in *Proc. of IEEE WoWMoM*, 2008.

[28] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using Redundancy to Cope with failures in a Delay Tolerant Network," in *Proc. of ACM SIGCOMM*, 2005.

[29] U. Lee, J. Park, J. Yeh, G. Pau, and M. Gerla, "Code Torrent: Content Distribution Using Network Coding in VANET," in *Proc. of ACM MOBISHARE*, 2006.

[30] Y. Lin, B. Li, and B. Liang, "Stochastic Analysis of Network Coding in Epidemic Routing," *IEEE JSAC*, vol. 26, no. 5, pp. 794–808, 2008.

Appendix

Proof of Lemma 1: The first order derivative of $B_{ij}^{real}(s_i)$ can be calculated as:

$$\frac{d(B_{ij}^{real}(s_i))}{d(s_i)} = \frac{d(\int_0^{gs_i} u f_{U'_{ij}}(u)\mathrm{d}u + (1-\int_0^{gs_i} f_{U'_{ij}}(u)\mathrm{d}u)gs_i)}{d(s_i)}.$$

Due to the continuity requirement of Differentiation under the Integral Sign, we check the continuity of $f_{U'_{ij}}$ first.

$$f_{U'_{ij}}(u) = \frac{d(F_{U'_{ij}}(u))}{du} = d(Pr\{Y_{ij}=0\} + Pr\{Y_{ij}=1\} Pr\{X_{ij}^{(1)} \le u\} + Pr\{Y_{ij}=2\}Pr\{X_{ij}^{(1)} + X_{ij}^{(2)} \le u\} + \cdots + Pr\{Y_{ij}=k\}Pr\{X_{ij}^{(1)} + \cdots + X_{ij}^{(k)} \le u\} + \cdots)/du$$
$$= Pr\{Y_{ij}=0\}\delta(u) + \sum_{k=1}^{\infty}(Pr\{Y_{ij}=k\} f_{\sum_{h=1}^k X_{ij}^{(h)}}(u))$$

where $F_{U'_{ij}}(u)$ is the Cumulative Distribution Function of U'_{ij}, and $\delta(u)$ is an impulse function. Although $f_{U'_{ij}}(u)$ is not continuous, $\forall k \in \mathbf{N}, f_{\sum_{h=1}^k X_{ij}^{(h)}}(u)$ are continuous, which are the convolutions of continuous functions. Thus, the derivation of $B_{ij}^{real}(s_i)$ can be calculated as:

$$\frac{d(B_{ij}^{real}(s_i))}{d(s_i)} = \sum_{k=1}^{\infty} g^2 s_i Pr\{Y_{ij}=k\} f_{\sum_{h=1}^k X_{ij}^{(h)}}(gs_i) - (\sum_{k=1}^{\infty} Pr\{Y_{ij}=k\} f_{\sum_{h=1}^k X_{ij}^{(h)}}(gs_i))g^2 s_i + g(1 - \int_0^{gs_i} f_{U'_{ij}}(u)\mathrm{d}u) = g(1-\int_0^{gs_i} f_{U'_{ij}}(u)\mathrm{d}u) > 0$$

$$\frac{d^2(B_{ij}^{real}(s_i))}{d(s_i^2)} = -g^2(\sum_{k=1}^{\infty} Pr\{Y_{ij}=k\} f_{\sum_{h=1}^k X_{ij}^{(h)}}(gs_i)) < 0$$

which proves the lemma.