Computer Networks 83 (2015) 184–198

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

MobiCache: Cellular traffic offloading leveraging cooperative caching in mobile social networks

Sheng Zhang [a,*], Jie Wu [b], Zhuzhong Qian [a], Sanglu Lu [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China
[b] Department of Computer and Information Sciences, Temple University, USA

ARTICLE INFO

Article history:
Received 3 November 2014
Received in revised form 10 February 2015
Accepted 2 March 2015
Available online 20 March 2015

Keywords:
Cellular traffic offloading
Content floating
Cooperative caching
Mobile social networks

ABSTRACT

Offloading cellular traffic through mobile social networks has arisen as a promising way for relieving cellular networks. Prior studies mainly focused on caching data in a number of pre-selected helpers. However, such a strategy would fail when mobile users enter and leave the target area over time. In this paper, we examine the research decisions and design tradeoffs that arise when offloading cellular traffic in such a dynamic area of interest, referred to as a *MobiArea*, and we design an offloading framework, MobiCache, for maximizing cellular operators' revenues and minimizing the overhead imposed on mobile devices. On the user side, we propose a content floating-based cooperative caching strategy that caches data in geographical floating circles, instead of selected helpers in previous studies, to cope with the dynamics. A geographical routing scheme is designed for delivering data and queries towards floating circles. We also develop a cache replacement scheme to improve caching cost-effectiveness inside floating circles. On the operator side, query history and feedback are maintained for cellular operators to optimize framework parameters that maximize their revenues. Extensive trace-driven simulations show that, compared with a state-of-the-art scheme, MobiCache offloads up to 52% more traffic with 15% shorter delay and 6% less forwarding cost.

1. Introduction

The past few years have witnessed the explosive popularity of smart-phones and tablets. According to Cisco Visual Networking Index (VNI) [1], the mobile data traffic generated in 2014 was nearly 30 times the size of the entire global Internet in 2000, and global mobile traffic will grow at a compound annual growth rate of 57 percent from 2014 to 2019; in other words, it will increase 10-fold and reach 24.3 exabytes per month by 2019. This huge amount of data traffic, as a consequence, has degraded service quality and created immense pressure on the limited spectrum of cellular networks. For example, AT&T has already made several measures for throttling its customers [2]. One straightforward solution to tackling this problem would be deploying more base stations to expand cellular network capacity [3], which is limited, however, since it requires high financial input but gets low and diminishing returns. Addressing this problem needs some paradigm-altering approaches.

Due to the inherent proximity-based sharing ability of mobile devices and the delay-tolerant nature of many cellular contents, offloading cellular traffic through mobile social networks (MSNs) [4] has arisen recently as a promising method for relieving cellular networks [5–11]. When users are rewarded with proper incentives [9,10], it is very likely that they will be willing to wait for a predetermined delay to obtain some cellular contents, and to help store-carry-forward them. Traffic offloading then can be realized

* Corresponding author. Tel.: +86 25 83681369.
E-mail addresses: sheng@nju.edu.cn (S. Zhang), jiewu@temple.edu (J. Wu), qzz@nju.edu.cn (Z. Qian), sanglu@nju.edu.cn (S. Lu).

http://dx.doi.org/10.1016/j.comnet.2015.03.011
1389-1286/© 2015 Elsevier B.V. All rights reserved.

through caching: some users intentionally cache cellular contents, which can be used to satisfy future queries from other users.

An important concern of this paradigm is that it may incur a long delay. Obviously not all cellular contents could bear long delays. But there are a large body of contents (*e.g.*, optional software update, interesting videos) that *not only permit offloading, but for which it is essential to do so in order to relieve cellular networks and lower users' subscription fees*. Prior studies mainly focus on placing data items in several selected helpers [5,10,12], so that future queries can be responded to without signaling between cellular networks and users. However, if mobile users enter and leave the target area over time, these strategies would fail to maintain data availability, since they could not find an appropriate group of helpers that can stay and serve inside the target area for a sufficiently long period.

In this paper, we examine the research decisions and design tradeoffs that arise when offloading cellular traffic in such a dynamic area of interest, referred to as a *MobiArea*. Our goal is to maximize cellular operators' revenues and minimize the overhead imposed on mobile devices. We develop an offloading framework, MobiCache, which caches data in geographical floating circles instead of fixed nodes. When a user requests a data item, the cellular operator offers *a choice* to him/her: obtain the data either from the cellular network immediately, or from that data's floating circle within a predetermined system-specified delay.

Several challenges have to be addressed to realize our goals. How to minimize the computational overhead imposed on mobile devices to conserve their batteries? How to route data or queries towards geographical floating circles without incurring too much forwarding cost? When two users with limited buffers have a contact, how to perform cache replacement to improve cost-effectiveness? How to determine the positions, radii, and lifetimes of floating circles, so as to minimize forwarding cost involved in circles, as well as the total access delay over all potential requesters?

We investigate techniques to settle these challenges in MobiCache, which is generally composed of two parts: the user side and the operator side. On the user side of our system, we propose to cache in floating circles, to overcome dynamics and develop a ticket-based geographical routing scheme for delivering data items and queries; a cache replacement strategy is also designed to improve caching cost-effectiveness. On the operator side, we maintain query history and feedback information collected over time for every data item, and perform predictions and estimations based on the following principle: similar data attracts similar users. In doing so, the computation-intensive task, *i.e.*, parameter determination, is completed by cellular operators, which lowers the overhead imposed on mobile devices and also is convenient for operators to maximize their revenues.

Our framework is of course not a panacea. We admit that many challenges remain, *e.g.*, privacy and energy issues. Addressing these concerns is a direction of our future work. While our approach does not generalize to all scenarios, we hope that our proposal will provide some potential guidelines for future offloading design.

The remainder of this paper is organized as follows. We go over related work and our contributions in Section 2. We provide the system model and motivate our work in Section 3. Section 4 overviews the proposed framework. Sections 5 and 6 present the details of our framework. We discuss known issues and extensions in Section 7. Before concluding the paper in Section 9, we evaluate the performance of the framework in Section 8.

2. Related work and our contributions

There is a rich heritage of studies on cellular network offloading and delay tolerable networks (DTNs) that informed and inspired our design. We describe a subset of these related efforts below.

An MSN can be considered as a type of DTN, which lacks persistent end-to-end connectivity, due to low node density and/or unpredictable node mobility. To cope with the intermittent connectivity, epidemic routing [13] is proposed to deliver packets via flooding; however, this incurs an extremely high forwarding cost. Some later work [14,15] reduces this cost through intelligent relay selection. Intentional routing [16] translates an administrator-specified routing metric into per-packet utilities. Social features are exploited to facilitate DTN routing [17] or improve hypertext results [18]. A Global Positioning System (GPS)-assisted geographical routing protocol is developed in [19] for vehicular delay-tolerant networks.

Prior studies on cellular traffic offloading through device-to-device connections [20] can be categorized into two broad types, based on the hop count between cellular networks and users: single-hop [10] and multi-hop [6,8]. Most of them [6,8,10] mainly focus on selecting a group of users as bridges between cellular networks and users. Moreover, WiFi throughput prediction is exploited for vehicular Internet access in [7]. Auction-based incentive mechanisms for offloading are designed in [9]. Computation offloading is investigated in [11].

The problem of placing data copies in nodes with limited memories is investigated in [5,21,22] with different goals: maximizing the total size of offloaded data, minimizing total access cost, and maximizing the probability of retrieving a file that contains multiple erasure blocks, respectively. The publish/subscribe based caching is investigated in [12]. Content floating technique [23,24] maintains data items through flooding them inside their respective floating circles and discarding them outside their respective circles. Theoretical analyses show that the expected data availability can be sufficiently long, provided that a critical condition is met. A short survey on caching mechanisms for web, ad hoc networks and DTNs is presented in [25], which also provides some inspiring ideas for content caching and retrieval for vehicular delay-tolerant networks.

Comparatively, in this paper, we propose to cache in floating circles, instead of fixed users, to cope with the dynamics of the target area, and we develop MobiCache for cellular operators to maximize their revenues, and minimize overhead on mobile devices. Although the idea of content floating is not new, to the best of our knowledge, MobiCache is the first system that supports offloading in dynamic areas without any stationary nodes' assistance (*e.g.*, throwboxes [26,27]). More specifically, the contributions of this paper are summarized as follows. (1) MobiCache intentionally maintains data availability within geographical floating circles to overcome the dynamics; (2) we propose a ticket-based geographical routing scheme for delivering data items and queries; (3) a dynamic programming-based caching replacement policy is designed for improving caching cost-effectiveness; and (4) we provide a set of effective heuristics for operators to optimize framework parameters to maximize their revenues.

3. Background and motivations

This section overviews the assumptions and models used in this paper and motivates the offloading design of MobiCache based on two observations.

3.1. Models and assumptions

We denote by *MobiArea* the dynamic area of interest. Denote a *mobile user* that has emerged in the MobiArea as n_i. In this paper, mobile nodes, users, and devices will be interchangeably used. Users n_i and n_j can communicate with each other through a wireless interface (WiFi or Bluetooth) if and only if their distance is not larger than a fixed range. This range, and the physical size of each mobile user, are assumed to be negligibly small compared to the dimensions of the MobiArea.

As assumed in prior studies [5,12,21,22], most users are not willing to contribute all of their mobile storage to caching cellular data, thus, we denote the available buffer size of user n_i as B_i. We also assume that each user is equipped with a GPS to determine his position; this can be easily achieved, since almost all of the smart-phones and tablets nowadays have a rich set of sensors. We will discuss how to deal with errors in positioning and how to reduce energy consumption for positioning in Section 7.

Denote a data item as d_i, and its size as s_i. For simplicity, we assume that all necessary data transfer can be completed during any contact. If s_i is too large to be transferred during a contact, then we can divide d_i into several small segments, each of which follows the assumption. The segmentation overhead is discussed in Section 7.

User n_i is associated with an interest vector I_i, represented by a $M \times 1$ vector $[p_1^i, p_2^i, \ldots, p_M^i]^T$, where M denotes the size of the keywords universe and $p_h^i \in [0, 1]$ indicates the probability of this user to be interested in the hth keyword [28]. The sum $\sum_{h=1}^{M} p_h^i$ is normalized to 1. Similarly, data item d_j is associated with a characteristic vector C_j, represented by a $M \times 1$ vector $C_j = [e_1^j, e_2^j, \ldots, e_M^j]^T$, where $e_h^j \in [0, 1]$ indicates the extent to which the hth keyword can characterize this data. Also, $\sum_{h=1}^{M} e_h^j = 1$. Then, the probability that n_i is interested in d_j is defined as:

$$P_{ij} = I_i^T C_j = \sum_{h=1}^{M} p_h^i \cdot e_h^j. \tag{1}$$

There are already some incentive mechanisms [9,10], which motivate users to tolerate a predetermined tolerable delay or to help in forwarding data. We will not discuss the design of such mechanisms, as it is orthogonal to the focus of this paper. Notations are summarized in Fig. 1 for reference.

3.2. Motivations

3.2.1. Popular regions

We examine the Dartmouth trace [29] to better understand user mobility patterns. The trace consists of two parts: *access point (AP) locations* which provides the locations of 623 APs (only 507 of them are valid) on the Dartmouth campus, and *client-AP associations* which provides the association information of 6022 clients over a period of 703 days. After proper preprocessing on the raw data, we find that most of the association records emerge during the last one-fourth of the duration, and most of the clients only have a small number of associations over 703 days. Therefore, we only use the mobility trace of 78 clients collected from the 525th day to the 703th day.

Fig. 2 shows the locations of 340 APs that have associations with at least one of the 78 clients during that time period. The number of associations of an AP is indicated by the type of the corresponding marker that represents the AP. We see that, there are several extremely popular APs. These popular regions will facilitate MobiCache, as MobiCache would need to maintain data availability at geographical floating circles for future data queries.

3.2.2. Pricing gap

Since users have to tolerate some delay before getting data through offloading, this charge should be cut down to some extent. For example, Verizon charges a user roughly 8 USD for 1 GB data [30], *i.e.*, 0.8 cents for 1 MB data; if a user gets a data item via offloading, Verizon may charge it $0.4 s_i$ cents, where s_i is the size of the data in MB.

When a user contributes one forwarding of a data item, the cellular operator rewards the user with a few cents, which should at least compensate for the energy consumption in forwarding the data. The data transfer rate between mobile devices is assumed to be 1 MB/s (the typical rate of Bluetooth v1.2 [31]), thus, transmitting a data item of s_i MB lasts about s_i seconds. According US Energy Information Administration report [32], the average price of residential electricity in the US was about 12 cents per kW h in 2013. Based on measurements in [33], the energy consumption rate of a mobile phone during transmission is about 2 W. Therefore, an operator should pay $(12 \times (2 \times s_i))/(1000 \times 3600) \approx 6.67 s_i \times 10^{-6}$ cents to a user who forwards a data item with a size of s_i in MB one time.

We see that, cellular operators could still make profit, as long as the forwarding times of a data item is less than $(0.4 s_i)/(6.67 s_i \times 10^{-6}) \approx 59,970$, which is sufficiently large for mobile users to cooperatively cache data in geographical floating circles. As evidenced in Section 8, the average forwarding number of a data item is approximately

Notation	Definitions
n_i	a mobile user/node/device
B_i	available buffer size of user n_i
I_i	the interest vector of user n_i
s_j	the size of data d_j
C_j	the characteristic vector of data d_j
P_{ij}	the probability that n_i is interested in d_j
d_j	a data item
$\mathbf{c_j}$	the 2-D coordinates of the floating circle center of d_j
r_j	the radius of the floating circle of d_j
l_j	the lifetime of the floating circle of d_j
k_j	# of tickets for delivering d_j to its circle
k_{ij}	# of tickets for delivering n_j's request for d_i's circle

Fig. 1. Main notations are summarized for reference.

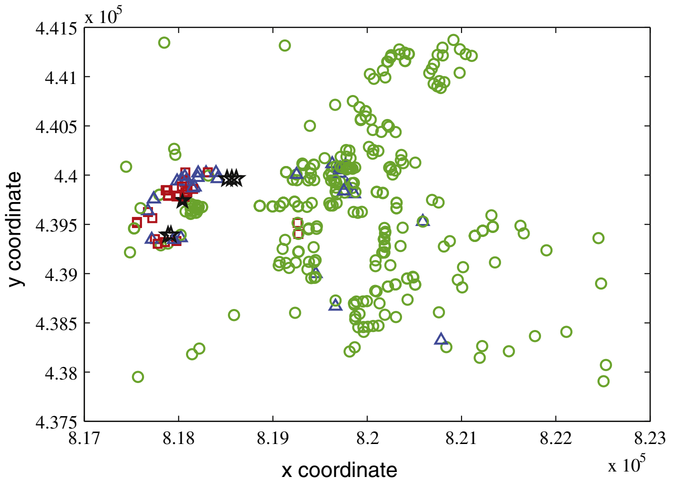


Fig. 2. The locations (indicated by the centers of markers) and the number of associations (circle: 0–100; triangle: 100–1000; square: 1000–10,000; star: ⩾10,000) of the selected 340 access points in Dartmouth trace [29].

250–800 in MobiCache, which is much smaller than the maximum allowable number of forwarding times. This tremendous gap allows us to design a cooperative caching-based method for efficient offloading.

4. MobiCache overview

Our objective is to offer a practical and feasible offloading service at dynamic areas without any stationary infrastructure assistance. We provide an overview of MobiCache through an illustrative example in Fig. 3. After user n_1 successfully downloads data d_1 via cellular networks, the cellular operator detects that this is the first download of d_1 in the MobiArea. Based on history and feedback information collected in the past (Section 6.1), the cellular operator learns that there may be many forthcoming requests for d_1; for example, the nodes with horizontal lines are potential requesters of d_1. To relieve the crowded cellular network but, meanwhile, satisfy those upcoming requesters, the cellular operator resorts to caching d_1 in a geographical floating circle.

4.1. On the cellular operator side

To maximize the cellular operator's revenues (Section 6.2), MobiCache estimates the following parameters that guide the set-up of the floating circle for d_1: (1) The coordinates of the circle center, $\mathbf{c_1} = (c_1^x, c_1^y)$ (Section 6.3), which should be determined to minimize the total access delay of all potential requesters of d_1. For example, there are more potential requesters of d_1 in the northwest part in this figure, so the circle center for d_1 is chosen near that part. (2) The radius of the floating circle, r_1 (Section 6.3). A large radius is beneficial to maintaining the availability of d_1, but incurs unnecessary forwarding cost. (3) The lifetime of the floating circle, l_1 (Section 6.4). As time goes on, the potential requesters of d_1 become fewer, while the maintenance cost becomes greater. MobiCache should carefully determine l_1 to maximize operators' revenue. When a floating circle expires, all devices involved in the circle are free to delete the corresponding data. (4) The maximum number of copies of d_1 when delivering d_1 to its circle, k_1 (Section 6.5). The

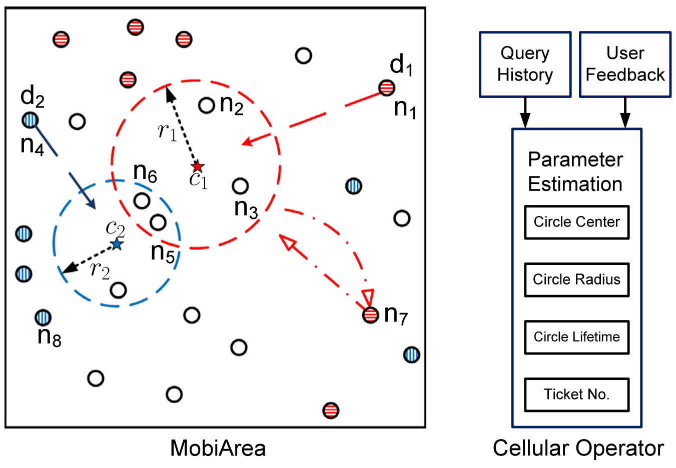


Fig. 3. The big picture of MobiCache. $\mathbf{c_i}$ and r_i specify the floating circle for d_i ($i = 1, 2$ in this example). The nodes with horizontal and vertical lines denote potential requesters of d_1 and d_2, respectively. The cellular operator estimates framework parameters based on query history and user feedback collected over time.

operator can use k_1 to control the tradeoff between the delivery cost (*i.e.*, the number of data replicates) and the delivery latency.

4.2. On the user side

Due to resource constraints of mobile devices, *e.g.*, battery lifetime and slow processing speed, MobiCache also tries to shift computation-intensive tasks to the cellular operator side, so as to reduce the computational overhead on mobile devices.

4.2.1. Delivering data to its floating circle

With proper incentive mechanisms, the cellular operator then sends these parameters to n_1 and asks n_1 to deliver d_1 to the specified circle. User n_1 employs the geographical routing scheme in Section 5.1 to deliver d_1 to its floating circle.

4.2.2. Maintaining data availability in floating circles

When the first copy of d_1 enters its circle, d_1 would get replicated whenever a node with it contacts another node without it inside the circle (Section 5.2). When a critical condition is met, the expected availability of d_1 can be sufficiently long, as demonstrated in [23,24]. The reason for using the circle shape for content floating is that a circle can be accurately expressed by its center and radius, yielding little additional information that needs to be transmitted.

4.2.3. Cache replacement policy for pairwise encounters

As there are some other data items, nodes with limited buffers may meet inside the intersection of two or more circles (*e.g.*, n_5 and n_6 in Fig. 3). A cache replacement strategy (Section 5.3) is then designed to improve cache cost-effectiveness.

4.2.4. Data query from users and response from floating circles

When node n_7 requests d_1 from cellular networks, the operator detects that there is already a floating circle for d_1, so the operator offers *a choice* to n_7: obtain the data either from the cellular network immediately, or from the floating circle of d_1 within a predetermined system-specified delay. If n_7 accepts the latter, the operator returns $\mathbf{c_1}$ and $k_{1,7}$ to n_7. Depending on the degree of interest, n_7 can freely choose either to physically move to the circle and get d_1, or to send queries (Section 5.4). If the latter happens, n_7 employs the geographical routing scheme in Section 5.1 with $k_{1,7}$ tickets to deliver queries to the circle of d_1, then waits for a response. *The operator also guarantees to push d_1 to n_7 via cellular networks once the system-specified delay passes.*

In the next two sections, we provide the design details of MobiCache on the user side and the cellular operator side, respectively.

5. MobiCache on the user side

In this section, we present the details of the proposed framework on the user side. Section 5.1 provides a routing scheme for a user sending a data item or query to a geographical circle. Section 5.2 introduces how to maintain data availability in floating circles. Section 5.3 deals with pairwise encounters within floating circles. Section 5.4 shows how to get data from floating circles. Most parameters involved in this section are estimated by cellular operators, which will be explained in Section 6.

5.1. Delivering data to a floating circle

Some prior studies [14–16] focused on delivering data packets to mobile destinations; different from them, we are interested in routing data items or queries to geographical regions. In the following, we propose a simple scheme based on active positioning. By "active positioning" we mean that, mobile users actively record their positions over time. In Section 7, we also show how to reduce energy consumption on positioning and cope with positioning errors through passive positioning.

In our framework, each user n_i records his position using GPS every γ_i time; we denote the position of n_i at time $k\gamma_i$ $(k = 0, 1, 2, \ldots)$ as $\mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}} = (posx^{n_i}_{k\gamma_i}, posy^{n_i}_{k\gamma_i})$. Here, γ_i is used to trade off between positioning frequency and energy consumption, *i.e.*, the user n_i can decrease γ_i to conserve energy. We estimate the moving speed $v(n_i)$ of n_i at time $k\gamma_i$ as

$$v(n_i) = \frac{\|\mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{(k-1)\gamma_i}}\|_2}{\gamma_i}. \quad (2)$$

As shown in Fig. 4, we estimate the moving direction of n_i at time $k\gamma_i$ as the direction of the vector $(\mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{(k-1)\gamma_i}})$. Define the deviation angle $\alpha(n_i)$ of n_i at time $k\gamma_i$ as the intersection angle of the moving direction, and the direction of $(\mathbf{c_d} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}})$, so $\alpha(n_i)$ is

$$\alpha(n_i) = arccos\frac{(\mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{(k-1)\gamma_i}}) \cdot (\mathbf{c_d} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}})}{\|\mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{(k-1)\gamma_i}}\|_2 \cdot \|\mathbf{c_d} - \mathbf{pos}^{\mathbf{n_i}}_{\mathbf{k\gamma_i}}\|_2}. \quad (3)$$

Based on this positioning information, we propose GAP, a GeogrAPhical routing scheme. Remember that, the cellular operator wants to minimize the overhead imposed on mobile devices, where the "overhead" regarding to routing is data forwarding and storage. To achieve this goal, the operator choose to restrict the number of forwardings and replications of a data item d within an integer k_d, which is also called the number of tickets in this paper. The operator can adjust k_d to reach a compromise between delivery delay and forwarding cost.

For each data item d, the initial ticket counter in its source node is k_d. When two nodes n_i and n_j have a contact, for each data d in the buffer of n_i, denote the ticket counters of d in n_i and n_j by t^d_i and t^d_j, respectively. Algorithm 1 shows the forwarding details in a contact.

Algorithm 1. The GAP scheme

```
 1: When two nodes $n_i$ and $n_j$ have a contact:
 2: **for** each data $d$ in the buffer of $n_i$ **do**
 3:   **if** $t^d_i \geqslant 1$ and $n_j$ waits for $d$ **do**
 4:     $n_i$ forwards $d$ to $n_j$
 5:   **if** $t^d_i > 1$ and $t^d_j = 0$ **do**
 6:     **if** $\alpha(n_i) \leqslant A$ and $\alpha(n_j) \leqslant A$ **do**
 7:       $n_i$ forwards $d$ to $n_j$
 8:       $t^d_j \leftarrow \left\lceil \frac{\alpha(n_i)+\epsilon}{\alpha(n_i)+\alpha(n_j)+2\epsilon} \cdot \frac{v(n_j)+\epsilon}{v(n_i)+v(n_j)+2\epsilon} \cdot t^d_i \right\rceil$
 9:       $t^d_i \leftarrow t^d_i - t^d_j$
10:     **if** $\alpha(n_i) > A$ and $\alpha(n_j) \leqslant A$ **do**
11:       $n_i$ forwards $d$ $n_j$
12:       $t^d_j \leftarrow t^d_i - 1$
13:       $t^d_i \leftarrow 1$
14: **end for**
```

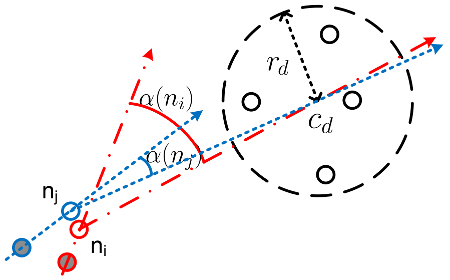


Fig. 4. The GAP scheme. When node n_i, with $c \geqslant 1$ ticket of data d, encounters node n_j without d, n_i then forwards data d and a certain number of tickets, determined by $\alpha(n_i), \alpha(n_j), v(n_i)$ and $v(n_j)$, to n_j.

If n_i has data d, and n_j waits for d, then n_i forwards d to n_j (lines 3–4); this happens when n_j wants to download d or n_j is an intermediate relay of d's query from another node. If $t^d_i > 1$ and $t^d_j = 0$, if neither of them deflect very much from the direction to the floating circle center (the deviation angle threshold A is set to $\pi/4$ in our trace-driven simulations), then the total tickets are split between them based on the ratios of their deviation angles and moving speeds (lines 6–9). Here, to avoid the case in which the dominator is zero, we add a sufficiently small positive ϵ to the formula. If the deviation angles of n_i and n_j are larger and not larger than the threshold A, respectively, the ticket counter of n_j is set to $t^d_i - 1$, *i.e.*, n_i leaves only one ticket for itself (lines 10–13). Any node that has d but with only one ticket is not allowed to replicate d unless it enters the floating circle of d. Any node could discard d if it leaves the floating circle of d, if it leaves the MobiArea, or if d expires.

We see from above that, using GAP, either of the number of forwardings and the number of copies of data d is restricted to k_d, allowing cellular operators to control the tradeoff between delivery cost and delay according to the statistics of the MobiArea.

5.2. Maintaining data availability in floating circles

Within the circle specified by $\mathbf{c_d}$ and r_d, data d is replicated whenever a node with it meets another node without it. Since such replication is based purely on the position of mobile nodes, every node in the circle should have a copy of d eventually. Nodes leaving the circle are free to delete their copies of d. Due to the random nature of users' movements, it seems that there is no guarantee for data availability. Surprisingly, as evidenced in [24], content floating can be quite reliable, as long as a *critical condition* is met; furthermore, the critical condition can be easily satisfied in normal circumstances. For example, when node mobility follows the random waypoint model [34], if we denote the transmission range of each node and the total number of nodes by *range* and *total*, respectively, and if the ratio h of floating circle radius to MobiArea radius tends to zero, the critical condition is

$$range \cdot total \cdot h > 16/45 \approx 0.356. \quad (4)$$

Some previous studies [12,21] keep data in several fixed popular nodes, while MobiCache stores data within geographical regions, which has the following advantage.

Mobile nodes enter and leave the MobiArea over time. Even if we cache data in some popular nodes, and let these nodes hand over their cached data to other nodes when they leave the MobiArea, what do we actually get? The answer is caching data in a region, and the region is, in fact, the entire MobiArea.

Some other studies [26,27] have proposed to strategically deploy throwbox, a kind of stationary wireless node that acts as a relay, to improve DTN throughput. Why not place some throwboxes in the MobiArea for caching? There are several concerns. First and foremost, throwboxes are expensive and cannot be massively deployed. Second, throwboxes restrict our choice of caching positions to a limited set of candidate locations—locations of throwboxes, which may result in performance degradation, indicated by the total access delay over all requesters.

5.3. Cache replacement policy for pairwise encounters

This section focuses on cache replacement for pairwise encounters inside floating circles, and *does not take into account data or query copies incurred by GAP (Section 5.1) or other routing protocols (Section 5.4)*. Ideally, a mobile user maintains a data item if this user is geographically inside the floating circle of this data item. However, user n_i has a limited buffer of size B_i; therefore, user n_i has to strategically select a subset of items for caching.

The cache replacement occurs when two mobile users with different sets of data items have a contact. In Fig. 5, user n_i meets another user n_j at time $k\gamma_i$; without loss of generality, the union set of data items cached at these two users is assumed to be $\mathbb{S} = \{d_1, d_2, \dots, d_m\}$. The cache replacement decision is based on *remaining lifetime* and *chord length*. The remaining lifetime rl_d of data d represents the remaining time before the floating circle of d expires. The chord length cl_{ki} of data d_k for a user n_i denotes the length that n_i would travel in the floating circle of d_k if n_i keeps its moving direction unchanged. Fig. 5 illustrates cl_{ki} and cl_{hi}. Given the center position and radius of a floating circle, the current position and the moving direction of a user, the chord length can be easily attained by plane geometry.

We note that the popularity of a data item is not considered here. The reason is implicit, if somewhat subtle: the lifetime (Section 6.4) of a floating circle already reflects the popularity of the respective data, *i.e.*, the floating circle of a more popular data has a longer lifetime.

We have the following argument: *both of n_i and n_j are inside the floating circle of every data item in $\mathbb{S}$*. Recall that, each user is free to delete a data item after he leaves the respective circle, which implies, if a node caches d_i, this node is inside the floating circle of d_i. In addition, as we assumed in Section 3.1, the transmission range and the physical size of every mobile user are negligibly small compared to the size of the MobiArea; taking these two facts together completes the argument.

Each data item has four options: cached at both nodes, at n_i, at n_j, or at neither of them, which makes the problem untractable. We propose to approximate it as follows: regarding $\mathbb{S}$ as the selection pool, n_i and n_j make their decisions separately. Taking n_i for example, we formulate the cache replacement as the following Knapsack problem [35]:

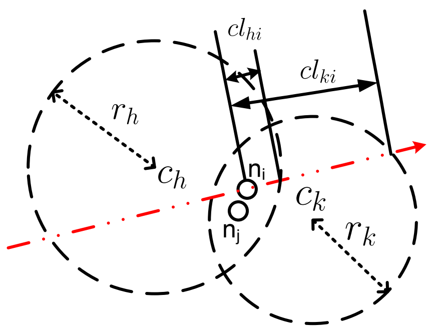


Fig. 5. Cache replacement. The weight of a data item d_k for user n_i is determined by the remaining lifetime rl_k before the floating circle of d_k expires and the chord length cl_{ki}.

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} x_k \cdot rl_k \cdot cl_{ki} \\ \text{subject to} \quad & \sum_{k=1}^{m} x_k \cdot s_k \leqslant B_i \\ & x_k \in \{0,1\}, \forall k \in \{1,2,\dots,m\} \end{aligned} \tag{5}$$

where x_k indicates whether d_k would be cached in n_i, and the product $(rl_k \cdot cl_{ki})$ serves as the weight of d_k for n_i. Observing that s_k and B_i are integers, we can use dynamic programming to solve this problem in $\mathcal{O}(mB_i)$ time.

5.4. Data query from users and response from floating circles

When a mobile user n_i requests d_k from the cellular operator, if the cellular operator detects that there is already a floating circle of d_k, the operator offers two choices: directly download the data from cellular networks, or indirectly get the data via opportunistic device-to-device connections within a predetermined delay. If n_i chooses the latter, the operator replies with $C_k, \mathbf{c_k}, r_k$, and k_{ki} (see Fig. 1 for notations). User n_i then checks the interest probability P_{ik} (see Eq. (1)). Depending on P_{ik}, n_i can choose either to physically move to the circle, or to send queries and wait for a response.

If n_i chooses to send queries, it employs GAP to send queries with k_{ki} tickets towards the floating circle of d_k. Any user n_j inside the circle of d_k exploits existing DTN routing algorithms,[1] *e.g.*, Spray and Wait [14] and delegation forwarding [15], for delivering d_k to n_i upon receiving the query. Before n_i receives any response from the floating circle, n_i might get a copy of d_k when contacting another user who happens to have a copy of d_k. Anyway, user n_i can send the request again to the cellular operator for free if he does not receive d_k after the predetermined system-specified delay passes, the operator then pushes d_k to n_i directly via cellular networks.

[1] As the physical position of n_i is not fixed, we cannot use GAP in this response process.

We note that, users may not be willing to decide, on a per-item basis, whether to use cellular networks or offloading. To make the procedure seamless, we allow users to setup their *policies* in our framework, *e.g.*, "download Facebook stuff as fast as possible, and optional software updates as cheaply as possible". In doing so, when users get data from floating circles, the service they experience is still seamless.

In this section, we see that the computation overhead at mobile devices is small. We shall see in the next section that, the computation-intensive task, *e.g.*, parameter estimation, is shifted to the cellular operator side.

6. MobiCache on the cellular operator side

This section presents the details of MobiCache on the operator side. We first introduce the query history and user feedback information maintained by cellular operators in Section 6.1, and presents the revenue maximization problem in Section 6.2. In Sections 6.3–6.5, we explain how to optimize framework parameters based on the collected information.

6.1. Query history and user feedback

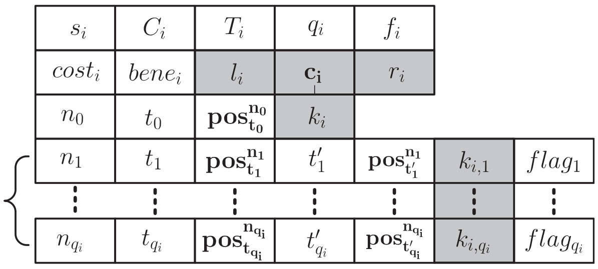
Fig. 6 shows the information maintained by a cellular operator for data d_i. In the first row, s_i is the size; C_i is the characteristic vector; T_i is the point of time when the cellular operator starts to set up the floating circle for d_i; q_i and f_i are the number of requests for d_i and the number of forwardings that d_i experiences from time T_i to $T_i + l_i$, respectively. In the second row, $cost_i$ and $bene_i$ will be introduced in the next subsection.

The third row records the first request for d_i: user n_0 sends a request for d_i at time t_0 at position $\mathbf{pos}_{\mathbf{t_0}}^{\mathbf{n_0}}$ and receives d_i via cellular networks; the cellular operator asks n_0 to deliver d_i to the respective floating circle with k_i tickets. Recall the definition of T_i, we have $T_i \approx t_0$.

Each following row corresponds to a request for d_i during the period from T_i to $T_i + l_i$: user n_j $(j = 1, 2, \ldots, q_i)$ sends a request for d_i at time t_j at position $\mathbf{pos}_{\mathbf{t_j}}^{\mathbf{n_j}}$, then the operator informs n_j to send a query to the respective circle with k_{ij} tickets, finally n_j receives d_i at time t'_j at position $\mathbf{pos}_{\mathbf{t'_j}}^{\mathbf{n_j}}$ via offloading $(flag_j = 1)$ or cellular networks $(flag_j = 0)$.

When users send requests or receive data, they are asked to provide feedback: geographical positions and the number of forwardings that they have helped in the past (including forwardings inside and outside floating circles). We mention that, feedback may violate the privacy of users; we do not consider privacy issues as they are out of the scope of this paper. In doing so, the operator obtains the white fields in Fig. 6; the rest of this section then focuses on determining the fields in shadow.

At the beginning, there is no query history or user feedback for the proposed framework to estimate framework parameters. We propose to *either use part of the trace as warm-up period to accumulate necessary information, or initialize the settings with empirical values.*



s_i	C_i	T_i	q_i	f_i		
$cost_i$	$bene_i$	l_i ▨	$\mathbf{c_i}$ ▨	r_i ▨		
n_0	t_0	$\mathbf{pos}_{\mathbf{t_0}}^{\mathbf{n_0}}$	k_i ▨			
n_1	t_1	$\mathbf{pos}_{\mathbf{t_1}}^{\mathbf{n_1}}$	t'_1	$\mathbf{pos}_{\mathbf{t'_1}}^{\mathbf{n_1}}$	$k_{i,1}$ ▨	$flag_1$
⋮	⋮	⋮	⋮	⋮	⋮ ▨	⋮
n_{q_i}	t_{q_i}	$\mathbf{pos}_{\mathbf{t_{q_i}}}^{\mathbf{n_{q_i}}}$	t'_{q_i}	$\mathbf{pos}_{\mathbf{t'_{q_i}}}^{\mathbf{n_{q_i}}}$	k_{i,q_i} ▨	$flag_{q_i}$

Fig. 6. The information maintained by the cellular operator for each data item d_i. Parameters in white grids can be obtained based on history and feedback; parameters in shadow grids need to be estimated.

6.2. Revenue maximization

Denote by $bene_i^j$ the benefit that the operator gains from delivering data d_i to user j; denote by $cost_i^j$ the reward that the operator offers to user j for forwarding data d_i one time. We note that, each data item d_i would be forwarded many times during the lifetime of its floating circle; the cellular operator tracks how many times a user forwards a data item through feedback, as shown in Fig. 6. We define them as follows:

$$bene_i^j = \begin{cases} 0, & \text{if the delivery is via cellular networks,} \\ 0.4 s_i, & \text{otherwise. } (s_i \text{ is the size of } d_i.) \end{cases} \tag{6}$$

$$cost_i^j = \begin{cases} 0, & \text{if the delivery is via cellular networks,} \\ 6.67 s_i \times 10^{-6}, & \text{otherwise.} \end{cases} \tag{7}$$

If the delivery is via cellular networks, we have $bene_i^j = cost_i^j = 0$; this is reasonable, since delivering d_i to user j via cellular networks implies that, cellular networks may fail to satisfy another user somewhere, due to its congestion. Otherwise, we set $bene_i^j$ and $cost_i^j$ according to the pricing gap in Section 3. We see that, for two different users n_j and n_k, $bene_i^j = bene_i^k$ and $cost_i^j = cost_i^k$, therefore, without causing any confusion, $bene_i^j$ and $bene_i$ will be used interchangeably, and the same holds for $cost_i^j$ and $cost_i$. The revenue of an operator is defined as follows:

$$\mathcal{R} = \sum_{d_i, n_j} (bene_i^j - cost_i^j). \tag{8}$$

To maximize $\mathcal{R}$, an operator should offload as much cellular traffic as possible, while incurring as little forwarding cost as possible. In the next three subsection, we present how to achieve this goal through parameter optimization, based upon query history and user feedback information.

6.3. Circle center and radius

The center position $\mathbf{c_i}$ of data d_i should be chosen to minimize the sum of distances between $\mathbf{c_i}$ and the positions of potential requesters while they are sending requests. But how can we know who would download d_i and where they are in advance? The basic principle is that, similar data items attract similar users. The degree of similarity [36] between d_j and d_i can be measured by the correlation coefficient ρ_{ji}:

$$\rho_{ji} = \frac{\sum_{k=1}^{M}(e_k^j - \bar{e^j})(e_k^i - \bar{e^i})}{\sqrt{\sum_{k=1}^{M}(e_k^j - \bar{e^j})^2}\sqrt{\sum_{k=1}^{M}(e_k^i - \bar{e^i})^2}}. \tag{9}$$

where $\bar{e^i}$ denotes the average of the elements in C_i. Given a threshold η_1, we define $\mathbb{S}^{d_i} = \{d_j | \rho_{ji} \geqslant \eta_1\}$, that is, the correlation coefficient between d_i and every item in $\mathbb{S}^{d_i}$ is not smaller than η_1. Then, $\mathbf{c_i}$ is estimated as the weighted average of the center positions of items in $\mathbb{S}^{d_i}$. That is,

$$\mathbf{c_i} = \sum_{d_x \in \mathbb{S}_i^d} \frac{\rho_{xi} \cdot \mathbf{c_x}}{\sum_{d_y \in \mathbb{S}_i^d} \rho_{yi}}. \tag{10}$$

For the radius r_i of data d_i, if it is chosen too small, then the critical condition in Eq. (4) cannot be satisfied; if too large, then some unnecessary forwardings and replications would be incurred. Hence, we set r_i to be the smallest value that satisfies the critical condition.

Please note that, $\mathbf{c_i}$ and r_i will be updated after the floating circle of d_i expires. This update can improve the accuracy of estimations for future data, since the updated values are realistic and accurate.

6.4. Circle lifetime

The circle lifetime l_i of data d_i should be chosen to maximize a cellular operator's revenue. The benefit of maintaining floating circles comes from serving data requests. The cost of maintaining floating circles results from data forwardings. Note that, when we need to estimate l_i, we do not have any information about requests and forwardings of d_i. Therefore, we need to estimate l_i based on information of some other data that is similar to d_i.

Since similar data attract similar users, we assume that, for each request for an item d_x in $\mathbb{S}^{d_i}$, there will be also a request for d_i, and the operator gains a benefit of $(\rho_{xi} \cdot bene_i)$ by satisfying this request. Remember that we have the start time and end time of each request; for example, in Fig. 7(a), t'_1 is the time when n_1 receives d_1 via device-to-device connections. For each $d_x \in \mathbb{S}^{d_i}$, denote by $\mathbb{N}_x^i(t)$ the set of users that receive d_x before time t. We use $b_i(t)$ to denote the total benefit of maintaining the circle of d_i when the lifetime of the circle is t. We have

$$b_i(t) = \sum_{d_x \in \mathbb{S}_i^d} \sum_{n_j \in \mathbb{N}_x^i(t)} \rho_{xi} \cdot bene_i^j. \tag{11}$$

Fig. 7 shows an example. We assume $s_3 = 1, \mathbb{S}^{d_3} = \{d_1, d_2\}, \rho_{13} = 0.3$, and $\rho_{23} = 0.8$. When $t = 3$, we have $\mathbb{N}_1^3(3) = \emptyset$ and $\mathbb{N}_2^3(3) = \{n_5\}$, then we have $b_3(3) = \rho_{23} \cdot bene_3 = 0.32$; when $t = 5$, we have $\mathbb{N}_1^3(5) = \{n_1\}$ and $\mathbb{N}_2^3(5) = \{n_5\}$, then we have $b_3(5) = \rho_{13} \cdot bene_3 + \rho_{23} \cdot bene_3 = 0.44$; and so on. We then have the $b_3(t)$ curve as in the Fig. 7(c).

The intuition for estimating the number of forwardings of d_i is that, floating circles with physically close centers have similar density of users. Given a threshold η_2, we define $\mathbb{S}_i^c = \{d_x | \, ||\mathbf{c_i} - \mathbf{c_x}||_2 \leqslant \eta_2\}$, that is, the distance between $\mathbf{c_i}$ and the floating circle center of any item in $\mathbb{S}_i^c$ is not larger than η_2. Then, the number of forwardings of d_i in unit time, denoted as Δc_i, can be estimated as

$$\Delta c_i = \frac{\sum_{d_x \in \mathbb{S}_i^c} f_x}{\sum_{d_x \in \mathbb{S}_i^c (l_x \cdot (r_x)^2)} \cdot (r_i)^2}. \tag{12}$$

We use $c_i(t)$ to denote the total cost of maintaining the circle of d_i when the lifetime of the circle is t. We have

$$c_i(t) = \Delta c_i \cdot cost_i \cdot t. \tag{13}$$

Fig. 7 shows an example, where Δc_3 is assumed to be 10^4. We have $c_3(t) = 0.0667t$ as in Fig. 7(c). Then, the lifetime l_3 is set to the value that maximize the revenue $b_3(t) - c_3(t)$. It is worth mentioning that, if l_i generated from our estimation is not positive, then the operator should not set up a floating circle for d_i, since it could not gain any revenue from offloading d_i.

6.5. Number of tickets in GAP

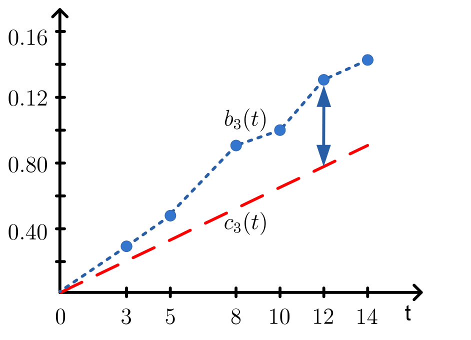
As we mentioned before, the ticket number is used to trade off between delivery delay and forwarding cost: if

n_1	t_1	$\mathbf{pos}_{\mathbf{t_1}}^{\mathbf{n_1}}$	$t'_1 = 5$
n_2	t_2	$\mathbf{pos}_{\mathbf{t_2}}^{\mathbf{n_2}}$	$t'_2 = 8$
n_3	t_3	$\mathbf{pos}_{\mathbf{t_3}}^{\mathbf{n_3}}$	$t'_3 = 10$
n_4	t_4	$\mathbf{pos}_{\mathbf{t_4}}^{\mathbf{n_4}}$	$t'_4 = 14$

(a) Request information of d_1

n_5	t_5	$\mathbf{pos}_{\mathbf{t_5}}^{\mathbf{n_5}}$	$t'_5 = 3$
n_6	t_6	$\mathbf{pos}_{\mathbf{t_6}}^{\mathbf{n_6}}$	$t'_6 = 8$
n_7	t_7	$\mathbf{pos}_{\mathbf{t_7}}^{\mathbf{n_7}}$	$t'_7 = 12$

(b) Request information of d_2



(c) The benefit $b_3(t)$ and cost $c_3(t)$ of d_3

Fig. 7. An example of estimating circle lifetime. We assume $\mathbb{S}^{d_3} = \{d_1, d_2\}, \rho_{13} = 0.3, \rho_{23} = 0.8, s_3 = 1$, and $\Delta c_3 = 10^4$. Then, we can plot $b_3(t)$ and $c_3(t)$ as in the figure, where the maximum marginal revenue happens when $t = 12$, thus, l_3 is set to 12.

the delivery delay is too long, some subsequent requests from other users may happen before the floating circle is set up; if it is too short, the floating circle may have to wait for a long time before any request arrives, which incurs unnecessary forwarding cost. It is hard to analyse the relationship between ticket number, delivery delay, and forwarding cost; furthermore, estimating the generation time of subsequent requests seems impossible. We therefore resort to the following heuristic.

6.5.1. Delivering a data item to a floating circle

This happens when setting up a floating circle. A data delivery can be characterized by its start and end positions. The degree of similarity between two data deliveries can be measured by the Euclidean distance between their start and end points. For example, user n_k is asked to send d_i towards $\mathbf{c_i}$ at time t_k; given a threshold η_3, we can select a set of data items: for each item d_x in this set, 1) the distance between $\mathbf{pos_{t_k}^{n_k}}$ and where the first requester of d_x sends a request for d_x, is not larger than η_3, 2) the distance between $\mathbf{c_x}$ and $\mathbf{c_i}$ is not larger than η_3. Denote this set as $\mathbb{S}_{n_j t_j}^{d_i}$, then the ticket number k_i can be estimated as

$$k_i = \left(\sum_{d_x \in \mathbb{S}_{n_j t_j}^{d_i}} k_x \right) \Big/ |\mathbb{S}_{n_j t_j}^{d_i}| \qquad (14)$$

where $|\mathbb{S}_{n_j t_j}^{d_i}|$ denotes the cardinality of the set. Note that, after the circle of d_i expires, k_i should be updated as follows: $k_i = k_i + 1$, if the floating circle is set up too late that some data requests have to retrieve data via cellular networks; otherwise, $k_i = k_i - 1$.

6.5.2. Delivering a data query to a floating circle

This happens when a user n_j sends a data request towards the floating circle of d_i. Using the same method as above, we can estimate k_{ij}. The only difference is that, data d_i has only one k_i, because the first requester of d_i is unique; but there are many k_{ij}'s, since there are many subsequent requesters. When user n_j eventually receives d_i, it informs the cellular operator of the delay he experienced. Then, k_{ij} is updated as follows: $k_{ij} = k_{ij} + 1$, if n_j obtains d_i via cellular networks; otherwise, $k_{ij} = k_{ij} - 1$.

The expositions hitherto focus on the scenario where one data item has only one floating circle. However, when there is some sort of extremely popular data that users wish to have in a very small delay, multiple floating circles could be set up within a MobiArea to reduce access delay. The main idea of MobiCache can adapt naturally to this new context.

7. Discussion

7.1. Segmentation overhead.

We have assumed that all necessary data transfer can be completed during any contact in Section 3.1. If a data item is too large to be transferred in one contact, then we can divide it into several small segments which follow the assumption. Let us take a look at the segmentation overhead. Denote by cc the communication capacity between two nodes, and by hd the length of the Bluetooth protocol header. For a data item d of size s, if $s + hd > cc$, then we divide it into $\lceil \frac{s}{cc-hd} \rceil$ segments. Remember that, the proposed framework uses ticket counters to restrict the number of replications of any item to be not larger than the number of tickets k (see Section 5.1). Therefore, the overall forwarding or storing overhead incurred by segmentation is not larger than $\lceil \frac{s}{cc-hd} \rceil \cdot hd \cdot k$. Though this value increases as the length of protocol header hd increases, in the current design of Bluetooth protocols, hd is far less than cc. Thus, the segmentation overhead is negligible.

7.2. Active vs. passive positioning

Location information is used in three aspects in the proposed framework. One is to help the framework determine whether the moving direction of a mobile user deflect much from the direction of a floating circle (Section 5.1); the second one is to help the framework determine whether a mobile user is within the floating circle of a data item (Sections 5.2 and 5.3); and the last one is to provide the location feedback when a user sends a query or receives a data item at the end (Section 5.4).

The current design of MobiCache uses active positioning, *i.e.*, mobile users actively record their positions over time. In fact, when there is energy concern, we can switch to passive positioning, *i.e.*, mobile users record their positions only when necessary. Specifically, for the first aspect above, we can use Spray and Wait [14] or Delegation Forwarding [15], both of which do not need geographical information, instead of GAP; for the second and third aspects, mobile users turn on their GPS only when encountering the others, sending data queries or finally receiving data items. We see that, this kind of passive positioning decreases the dependence on location information, and meanwhile conserve energy to some extent.

Besides, when there are positioning errors, for the second aspect above, we just need to set the radius of a floating circle to be the one calculated in Section 6.3 plus the maximum positioning error, which can guarantee that the critical condition in Eq. (4) is still met. For the third aspect, position errors would not be amplified when deciding the center of a circle. Therefore, passive positioning can mitigate the negative impact of position errors.

8. Performance evaluation

In this section, we conduct extensive trace-driven simulations to evaluate our design under different network settings and reveal insights of the proposed design performance.

8.1. Realistic traces and baseline algorithms

Our evaluations are conducted on three realistic traces, where Bluetooth-enables devices (*e.g.*, iMotes) periodically detects their peers nearly and record the contacts over several days/months. Fig. 8 provides a brief summary of them. Compared with the Dartmouth trace, the Cambridge and Infocom06 traces are relatively small in terms of number of pairwise contacts. We choose them in our simulations, as we want to examine the performance of the proposed framework in scenarios of different scales.

There are many previous studies on mobile traffic offloading [13,10,7,23,24]. Wiffler [7] focused on WiFi throughput prediction for vehicular Internet access. Content floating technique was proposed and evaluated in [23,24], which serves as a basic component in MobiCache. Therefore, we compare the performance of our framework with *Flooding* [13], where data items are replicated whenever possible, and *VIP-Delegation* [10], where helper nodes are carefully selected as bridges between regular users and cellular networks.

8.2. Experiment setup

The buffer size B_i of user n_i is uniformly generated from $0.5B_{avg}$ to $1.5B_{avg}$, where B_{avg} is used to simulate different buffer constraints. We adopt a similar approach as that in [12] to generate interest vectors of mobile users: the probability p_h^i of user n_i to be interested in the h^{th} keyword is randomly drawn from a normal distribution with P_h as the mean value. P_h follows the Zipf distribution, which is the most famous model of web requests, that is, $P_h = (1/h^s)/(\sum_{i=1}^{M} 1/i^s)$, and we set the exponent s to be 2.

The cellular operator periodically generates a new data item; the generation interval T_{gen} is used to achieve different offloading loads. The size of each data item is uniformly distributed in $[20MB, 60MB]$. There are 10 keywords in the universe. The extent e_h^i to which the hth keyword can characterize data n_i is randomly chosen between 0 and 1, then we multiply each extent by a positive value, so as to normalize the sum of these extents to be 1. $bene_i$ and $cost_i$ are set to $0.4s_i$ and $6.67s_i \times 10^{-6}$ cents, respectively.

Each user checks whether the operator has generated new data every $T_{gen}/2$ time, and decides to download a new data item via the underlying MSN if his interest probability is larger than a threshold. The threshold is determined as follows: we generate a large number of data items in advance and calculate the interest probabilities, then the threshold is chosen as the average probability. In doing so, each user would only download about a half of the generated data items. The predetermined delay that mobile users have to wait for before obtaining mobile data is generated from $[0.5T_{delay}, 1.5T_{delay}]$.

We assume that each user in three traces follows a random waypoint model. Then, the radius of a floating circle can be determined by Eq. (4). Take the Dartmouth trace for example, the communication range is set to 100, the number of users is 78, and the radius of the movement circle is about 1732.75. (All the APs are deployed within a rectangle of 5087.84×3465.5 according to [29].) In our simulations, we add an additional length ($R_{plus} \times 1732.75$) to the radius calculated by Eq. (4), then see the impact of R_{plus}.

If a user chooses to physically move to a floating circle, we assume this user spends some time, which is determined by the distance and the average moving speed, on approaching the circle. Then, the user moves to an appropriate position according to his own mobility records. We use Spray and Wait [14] as the routing protocol for data response.

At the beginning, we set the default ticket number in GAP to be 10, 10, and 30, the default circle lifetime to be 1 day, 1 day, and 20 days in three traces, respectively, and the default floating circle center to be the location of the most popular iMote or AP. By default, the average buffer size B_{avg} is set to 120 MB, the radius coefficient R_{plus} is set to 1/8, the average tolerant delay T_{delay} is set to 20 days, and cache replacement policy is performed.

Trace	Cambridge [37]	Infocom06 [38]	Dartmouth [29]
Duration (days)	∼5	∼4	∼179
# of devices	12	78	78
# of contacts	4,229	128,979	421,875

Fig. 8. Realistic traces summary. (See above-mentioned references for further information.)

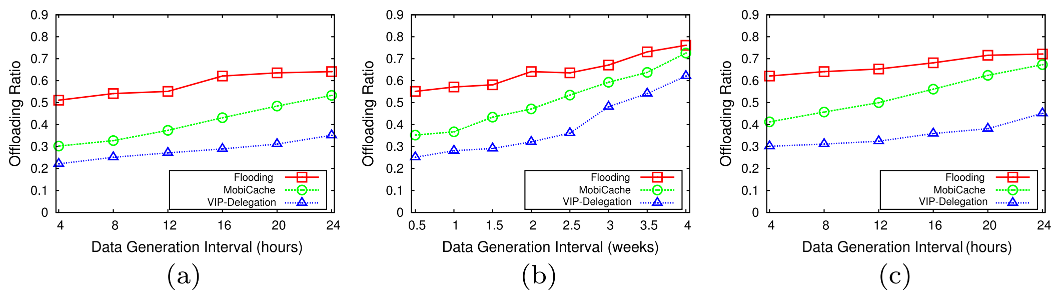


Fig. 9. Offloading ratio comparison on three traces. (a) Cambridge trace. (b) Dartmouth trace. (c) Infocom06 trace.

8.3. Performance comparison

(1) Offloading ratio: We evaluate the performance of the proposed framework under varying data generation intervals on three traces. Fig. 9 shows the results. We notice that, when the data generation interval T_{gen} increases, the offloading ratios of all three algorithms go up. In every trace, Flooding has the largest ratio, and MobiCache outperforms VIP-Delegation. On average, MobiCache offloads up to 45%, 31%, and 52% more cellular traffic than VIP-Delegation on Cambridge, Dartmouth, and Infocom06 traces, respectively. We also observe that, the average offloading ratio of MobiCache on Infocom06 trace is averagely higher than those on the other two traces. The main reason is that, Infocom06 trace has the highest contact density, *i.e.*, the number of contacts per day in three traces are 845.8, 2356.8, and 32,245, respectively. Thus, mobile nodes in Infocom06 trace has a much higher chance to meet the other nodes, making data more easily spread across the network.

(2) Offloading delay: Fig. 10 shows the average offloading delays of different algorithms on three traces. MobiCache produces a up to 15%, 10%, and 14% shorter delay than VIP-Delegation on Cambridge, Infocom06, and Dartmouth traces, respectively. The average offloading delays of three algorithms in Dartmouth trace are much larger than those in the other two traces. The main reason is that, the original duration of Dartmouth trace is around 703 days, and we extract only the contacts in 179 days from the trace. Our extraction may cause the network to be sparse, and thus leads to a longer delay.

(3) Forwarding overhead: Fig. 11 presents the forwarding overhead of three algorithms. Data items are replicated whenever there is a contact in Flooding, thus, it incurs the highest forwarding overhead among three algorithms, which is extremely obvious in Dartmouth trace. It is worthwhile to note that MobiCache results in 20% more, 6% less, and 6% less overhead than VIP-Delegation in Cambridge, Dartmouth, and Infocom06 traces, respectively. This is because that, the Cambridge trace is relatively small in terms of number of devices and contacts, thus, the helper nodes that are carefully chosen according to the betweenness centrality are enough to handle offloading in our settings.

8.4. Sensitivity analysis

(1) Impact of buffer size: We notice in Fig. 12(a) and (b), when T_{gen} goes up, fewer data items are generated; hence, the offloading ratio increases, and the forwarding number decreases. In particular, we note in Fig. 12(a) that, MobiCache with $B_{avg} = 240$ MB is about 40% better than that with $B_{avg} = 80$ MB when $T_{gen} = 0.5$ week, and 5% better when $T_{gen} = 4$ weeks. The reason behind this phenomenon is that when there is a small number of data items to be offloaded, the importance of buffer size abates. In Fig. 12(b), there are only slight differences of forwarding cost among different settings.

2) Impact of radius coefficient: In Fig. 12(c) and (d), the rising of R_{plus} increases the offloading ratio and forwarding number. Specifically, we see in Fig. 12(c) that, even we add an additional length of $1/4 \times 1732.75$ to the radius obtained by Eq. (4), the offloading ratio rises a little. This means that, the data availability is well guaranteed as long as the critical condition is met. In Fig. 12(d), as R_{plus} goes up, the number of forwardings increases, since more users would be involved in content floating circles.

3) Impact of cache replacement: In Fig. 12(e), our replacement policy improves the offloading ratio, and the improvement is particularly significant when T_{gen} is small. An interesting but not surprising observation is that, the average offloading delay of MobiCache with replacement is longer than that without replacement, as shown in Fig. 12(f). The reason is that, when there is no replacement, data availability is impaired, then some requesters may fail to receive responses within their tolerable delays. The

Trace	Flooding	MobiCache	VIP-Delegation
Cambridge ($T_{gen} = 4$)	20.57	27.65	32.45
Infocom06 ($T_{gen} = 4$)	19.46	25.43	28.34
Dartmouth ($T_{gen} = 336$)	343.68	652.56	755.28

Fig. 10. Offloading delay comparison (hours).

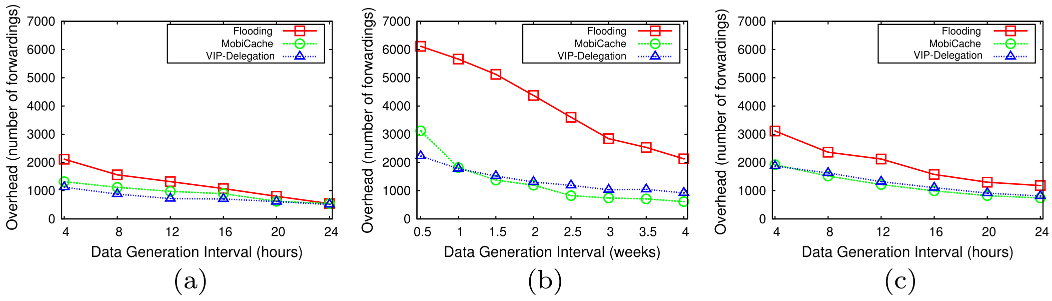
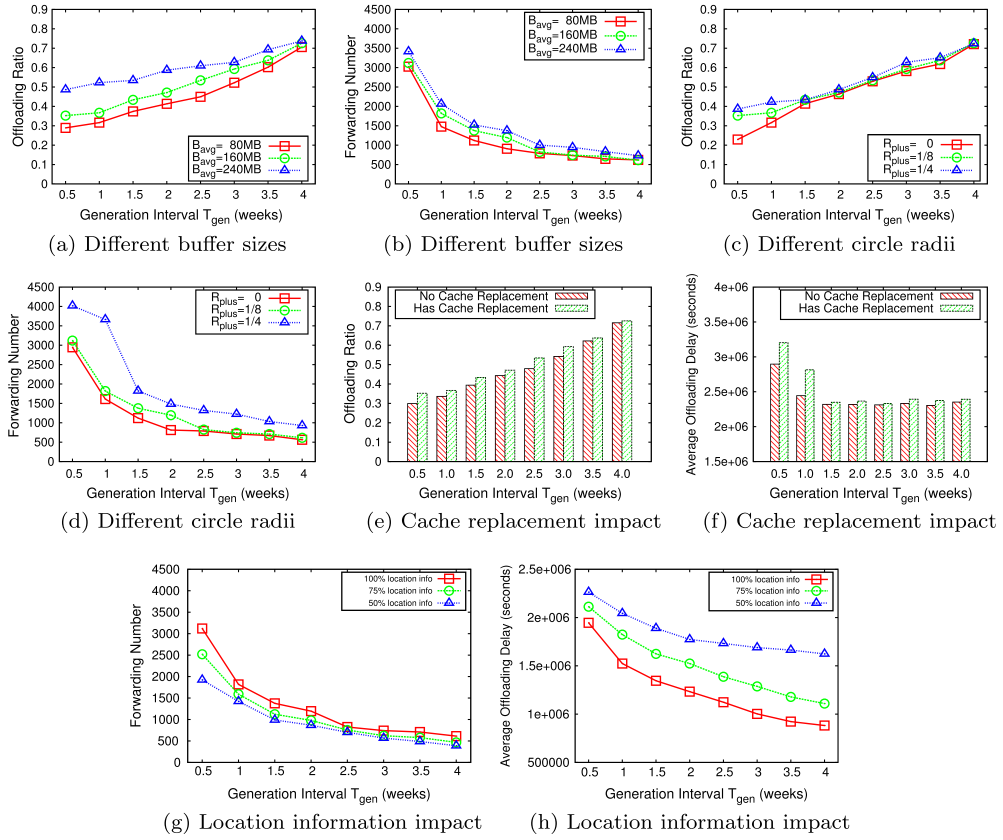


Fig. 11. Overhead (in terms of number of forwardings) comparison on three traces. (a) Cambridge trace. (b) Dartmouth trace. (c) Infocom06 trace.



(a) Different buffer sizes (b) Different buffer sizes (c) Different circle radii (d) Different circle radii (e) Cache replacement impact (f) Cache replacement impact (g) Location information impact (h) Location information impact

Fig. 12. Sensitivity analysis of MobiCache on Dartmouth trace.

delays these requesters experience are not included in the average offloading delay metric, because they finally download data through their cellular accesses.

(4) Impact of location information: Due to energy conservation or line-of-sight block from buildings, GPS location information may not be available all the time, *i.e.*, sometimes we have only partial location information. In our simulations, we intentionally remove a part of this information, and see the impact on forwarding cost and offloading delay. Fig. 12(g) and (h) shows the results. We observe that, when the percentage of available location information decreases, during some pairwise encounters, data cannot be replicated, since some nodes do not have location information and they cannot determine whether they are within a floating circle, or whether their deviation angles are less than the deviation threshold A or not. In this case, the number of forwardings should decrease as shown in Fig. 12(g). It is also not hard to see that, partial location information also negatively affects maintaining data availability in floating circles and data/query delivery delay, thus, the overall offloading delay of the proposed framework also increases as the percentage of location information decreases.

9. Conclusions

This paper introduces our initial effort in offloading cellular traffic in a MobiArea without any assistance from stationary nodes, and we propose the MobiCache framework, consisting of the GAP routing scheme, content caching in floating circles, cache replacement policy, and parameter estimations. We show through extensive trace-driven evaluations that, MobiCache offloads up to 52% more traffic with 15% shorter delay and 6% less forwarding cost than a state-of-the-art scheme. Our future work will focus on two directions. One direction focuses on exploiting social features of mobile users in offloading cellular traffic, while the other direction involves extending our framework to more predictable environments.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful suggestions. This work was supported in part by NSFC (61321491 and 91218302), Key Project of Jiangsu Research Program (BE2013116), EU FP7 IRSES MobileCloud Project (612212), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

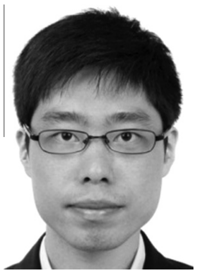
References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014–2019. <http://u2l.info/1NoUTo>. (viewed 05.02.15).
[2] AT&T Throttling Their Smartphone Customers to Deal with Spectrum Shortage. <http://u2l.info/1nkLbB> (viewed June 16.06.14).

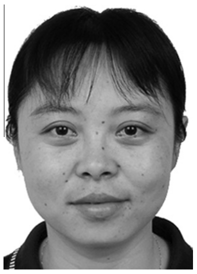
[3] V. Chandrasekhar, J. Andrews, A. Gatherer, Femtocell networks: a survey, IEEE Commun. Mag. 46 (9) (2008) 59–67.
[4] M. Motani, V. Srinivasan, P.S. Nuggehalli, Peoplenet: engineering a wireless virtual social network, in: Proc. of ACM MobiCom 2005, pp. 243–257.
[5] Y. Li, M. Qian, D. Jin, P. Hui, Z. Wang, S. Chen, Multiple mobile data offloading through disruption tolerant networks, IEEE Trans. Mob. Comput. 13 (7) (2014) 1579–1596.
[6] J. Whitbeck, Y. Lopez, J. Leguay, V. Conan, M.D. De Amorim, Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding, Elsevier Pervasive Mob. Comput. 8 (5) (2012) 682–697.
[7] A. Balasubramanian, R. Mahajan, A. Venkataramani, Augmenting mobile 3G using WiFi: measurement, system design, and implementation, in: Proc. of ACM MobiSys 2010, pp. 209–222.
[8] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, A. Srinivasan, Mobile data offloading through opportunistic communications and social participation, IEEE Trans. Mob. Comput. 11 (5) (2012) 821–834.
[9] X. Zhuo, W. Gao, G. Cao, S. Hua, An incentive framework for cellular traffic offloading, IEEE Trans. Mob. Comput. 13 (3) (2014) 541–555.
[10] M. Barbera, J. Stefa, A. Viana, M. de Amorim, M. Boc, VIP delegation: enabling VIPs to offload data in wireless social mobile networks, in: Proc. of IEEE DCOSS 2011, pp. 1–8.
[11] S. Zhang, J. Wu, S. Lu, Minimum makespan workload dissemination in DTNs: making full utilization of computational surplus around, in: Proc. of ACM MobiHoc 2013, pp. 293–296.
[12] W. Gao, G. Cao, A. Iyengar, M. Srivatsa, Supporting cooperative caching in disruption tolerant networks, in: Proc. of IEEE ICDCS 2011, pp. 151–161.
[13] A. Vahdat, D. Becker, Epidemic Routing for Partially-Connected Ad Hoc Networks, Duke University, Tech. Rep. 2000.
[14] T. Spyropoulos, K. Psounis, C.S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, in: Proc. of ACM SIGCOMM WDTN 2005, pp. 252–259.
[15] V. Erramilli, M. Crovella, A. Chaintreau, C. Diot, Delegation forwarding, in: Proc. of ACM MobiHoc 2008, pp. 251–260.
[16] A. Balasubramanian, B. Levine, A. Venkataramani, DTN routing as a resource allocation problem, in: Proc. of ACM SIGCOMM 2007, pp. 373–384.
[17] J. Wu, Y. Wang, Hypercube-based multipath social feature routing in human contact networks, IEEE Trans. Comput. 63 (2) (2014) 383–396.
[18] J.E. Costa, J.J. Rodrigues, T.M. Simões, J. Lloret, Exploring social networks and improving hypertext results for cloud solutions, Mob. Networks Appl. (2014) 1–7.
[19] V.N. Soares, J.J. Rodrigues, F. Farahmand, GeoSpray: a geographic routing protocol for vehicular delay-tolerant networks, Inform. Fusion 15 (2014) 102–113.
[20] S. Andreev, A. Pyattaev, K. Johnsson, O. Galinina, Y. Koucheryavy, Cellular traffic offloading onto network-assisted device-to-device connections, IEEE Commun. Mag. 52 (4) (2014) 20–31.
[21] B. Tang, H. Gupta, S.R. Das, Benefit-based data caching in ad hoc networks, IEEE Trans. Mob. Comput. 7 (3) (2008) 289–304.
[22] Y. Huang, Y. Gao, K. Nahrstedt, W. He, Optimizing file retrieval in delay-tolerant content distribution community, in: Proc. of ICDCS 2009, pp. 308–316.
[23] J. Kangasharju, J. Ott, O. Karkulahti, Floating content: Information availability in urban environments, in: Proc. of IEEE PERCOM Workshops 2010, pp. 804–808.
[24] E. Hyytia, J. Virtamo, P. Lassila, J. Kangasharju, J. Ott, When does content float? characterizing availability of anchored information in opportunistic content sharing, in: Proc. of IEEE INFOCOM 2011, pp. 3137–3145.
[25] B. Silva, V.N. Soares, J.J. Rodrigues, Towards intelligent caching and retrieval mechanisms for upcoming proposals on vehicular delay-tolerant networks, J. Commun. Software Syst. 7(1).
[26] W. Zhao, Y. Chen, M.H. Ammar, M.D. Corner, B.N. Levine, E.W. Zegura, Capacity enhancement using throwboxes in DTNs, in: Proc. of IEEE MASS 2006, pp. 31–40.
[27] N. Banerjee, M.D. Corner, D. Towsley, B.N. Levine, Relays, base stations, and meshes: enhancing mobile networks with infrastructure, in: Proc. of ACM MobiCom 2008, pp. 81–91.
[28] W. Gao, G. Cao, User-centric data dissemination in disruption tolerant networks, in: Proc. of IEEE INFOCOM 2011, pp. 3119–3127.
[29] T. Henderson, D. Kotz, I. Abyzov, The changing usage of a mature campus-wide wireless network, Comput. Netw. 52 (14) (2008) 2690–2712.
[30] Verizon Plans. <http://www.verizonwireless.com> (viewed 16.06.14).
[31] J. Haartsen, M. Naghshineh, J. Inouye, O.J. Joeressen, W. Allen, Bluetooth: vision, goals, and architecture, ACM SIGMOBILE Mob. Comput. Commun. Rev. 2 (4) (1998) 38–45.
[32] US EIA. <http://www.eia.gov> (viewed 16.06.14).
[33] Y. Xiao, R.S. Kalyanaraman, A. Yla-Jaaski, Energy consumption of mobile youtube: quantitative measurement and analysis, in: Proc. of IEEE NGMAST 2008, pp. 61–69.
[34] W. Navidi, T. Camp, Stationary distributions for the random waypoint mobility model, IEEE Trans. Mob. Comput. 3 (1) (2004) 99–108.
[35] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, 2001.
[36] S. Dowdy, S. Wearden, Statistics for Research, The Wiley Press, 2011.
[37] J. Scott, J. Crowcroft, P. Hui, C. Diot, Haggle: a networking architecture designed around mobile users, in: Proc. of WONS 2006, pp. 78–86.
[38] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, J. Scott, Impact of human mobility on opportunistic forwarding algorithms, IEEE Trans. Mob. Comput. 6 (6) (2007) 606–620.

Sheng Zhang received his BS and Ph.D. degrees from Nanjing University in 2008 and 2014, respectively. Currently, He is an assistant researcher in the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Laboratory for Novel Software Technology. His research interests include network virtualization, cloud computing, and mobile networks. To date, he has published more than 15 papers, including those appeared in IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, ACM MobiHoc, and IEEE INFOCOM. He received the Best Paper Runner-Up Award from IEEE MASS 2012.

Jie Wu is the chair and a Laura H. Carnell professor in the Department of Computer and Information Sciences at Temple University. He is also an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University. Prior to joining Temple University, he was a program director at the National Science Foundation and was a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He is regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. He was general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, and IEEE ICDCS 2013, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. Currently, he is serving as general chair for ACM MobiHoc 2014. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.



Zhuzhong Qian is an associate professor at the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Laboratory for Novel Software Technology. He received his Ph.D. Degree in computer science in 2007. Currently, his research interests include cloud computing, distributed systems, and pervasive computing. He is the chief member of several national research projects on cloud computing and pervasive computing. He has published more than 30 research papers in related fields. He is a member of CCF and IEEE.



Sanglu Lu received her Ph.D. degree in computer science from Nanjing University in 1997. She is currently a professor in the Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks, and pervasive computing. She has published over 80 papers in referred journals and conferences in the above areas. She is a member of CCF and IEEE.