Delay-Constrained Caching in Cognitive Radio Networks

Jing Zhao, *Student Member, IEEE*, Wei Gao, *Member, IEEE*, Yi Wang, and Guohong Cao, *Fellow, IEEE*

Abstract—In cognitive radio networks, unlicensed users can use under-utilized licensed spectrum to achieve substantial performance improvement. To avoid interference with licensed users, unlicensed users must vacate the spectrum when it is accessed by licensed (primary) users. Since it takes some time for unlicensed users to switch to other available channels, the ongoing data transmissions may have to be interrupted and the transmission delay can be significantly increased. This makes it hard for cognitive radio networks to meet the delay constraints of many applications. In this paper, we develop caching techniques to address this problem. We formulate the cache placement problem in cognitive radio networks as an optimization problem, where the goal is to minimize the total cost, subject to some delay constraint, i.e., the data access delay can be statistically bounded. To solve this problem, we propose a cost-based approach to minimize the caching cost, and design a delay-based approach to satisfy the delay constraint. Then, we combine them and propose a distributed hybrid approach to minimize the caching cost subject to the delay constraint. Simulation results show that our approaches outperform existing caching solutions in terms of total cost and delay constraint, and the hybrid approach performs the best among the approaches satisfying the delay constraint.

Index Terms—Cognitive radio networks, data access, cache placement, delay constraint

1 Introduction

Due to the proliferation of unlicensed wireless devices, unlicensed spectrum (e.g., ISM) is becoming increasingly congested; On the other hand, some licensed spectrum (e.g., UHF) is highly under-utilized. As a result, FCC approved unlicensed use of licensed spectrum through cognitive radio techniques [1], which enable dynamic configuration of the operating spectrum.

To avoid interference with licensed users, unlicensed users must vacate the spectrum when it is accessed by the *primary users* who are licensed to access the spectrum. Since it takes some time for the unlicensed users to switch to other available channels, the ongoing data transmissions may have to be interrupted, and the transmission delay will be significantly increased [2]. As a result, it is hard to meet the delay constraints of many applications in cognitive radio networks.

One way to reduce the data access delay is through caching. Suppose node v wants to access the data generated at node u which is faraway. Without caching, the data has to travel multiple hops to reach v. If any link along the routing path is affected by the primary user appearance, the data transmission will be interrupted, which increases the data access delay. If u's data is cached/replicated at multiple nodes in the network, v will be able to access the data from nearby nodes. This reduces the chance of transmission interruption, and then reduces the data access delay.

Although caching techniques have been well studied in traditional wireless networks, they cannot be directly applied to cognitive radio networks due to the following reasons. Traditional caching techniques assume the link transmission delay is known. For example, some previous works [3], [4], [5] model the access delay by the number of hops required to obtain the data, and assume all links have equal transmission delay. Some others [6], [7] assume the link transmission delay can be calculated accurately. However, in cognitive radio networks, the link transmission delay may vary significantly from time to time due to the primary user appearance. This makes it difficult to determine appropriate caching nodes to reduce the data access delay. Although the data access delay can be reduced by caching data at more nodes, this approach may not be efficient due to the caching cost which is affected by many factors. For example, to maintain cache consistency, disseminating data to more caching nodes consumes more energy and bandwidth, especially when the data is frequently updated. For each node, it also incurs some cost to access the data. The primary user appearance makes it difficult to estimate various types of cost, and further complicates the design of caching algorithms.

To overcome the aforementioned difficulties, we model the primary user appearance as a continuous-time Markov chain following several prior works [8], [9], and then derive the distribution of the link transmission delay and the data access delay. Since the data access delay depends on the primary user appearance, it may be long in some cases. By caching data at the right places, we can statistically control the data access delay within the *delay constraint* [10], [11], i.e., the data access delay is statistically bounded. To the best of our knowledge, this is the first paper to study caching techniques in cognitive radio networks for statistically

- *J. Zhao, Y. Wang, and G. Cao are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802. E-mail: {juz139, yuw124, gcao}@cse.psu.edu.*
- *W. Gao is with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996. E-mail: weigao@utk.edu.*

Manuscript received 14 July 2014; revised 14 Mar. 2015; accepted 20 Apr. 2014. Date of publication 27 Apr. 2015; date of current version 2 Feb. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2426715

controlling the data access delay. The detailed contributions of the paper are three folds.

- We develop caching techniques to meet the delay constraints of data access in cognitive radio networks. We formulate the cache placement problem in cognitive radio networks as to minimize the total cost, subject to the delay constraint.
- By modeling the primary user appearance as a continuous-time Markov chain, we derive the distribution of the link transmission delay and the data access delay.
- We propose three caching approaches: cost-based, delay-based, and hybrid, to solve the cache placement problem. Simulation results show that our approaches outperform existing caching solutions in terms of total cost and delay constraint, and the hybrid approach has the lowest total cost among the approaches satisfying the delay constraint.

The remainder of the paper is organized as follows. Section 2 reviews related work. In Section 3, we provide an overview of our work. We define the system models in Section 4, and then formulate our problem in Section 5. Sections 6, 7 and 8 present the three caching approaches in detail. We show simulation results in Section 9, and conclude the paper in Section 10.

2 Related Work

Caching is an effective technique to reduce the data access delay in wireless networks. The problem of finding the optimal cache placement can be formalized as a special case of the *connected facility location problem* [12], which is known to be NP-hard. The problem can be solved by using a greedy algorithm *poach* [4], which is within a factor of 6 of the optimal solution. If there are multiple data items, a node (user) may not cache all of them due to the capacity constraint. To address this challenge, cooperative caching [3], [6], [7], [13], [14] allows the sharing and coordination of cached data among multiple nodes. However, these caching approaches cannot be applied to cognitive radio networks since they assume the link transmission delay is known, but the link transmission delay in cognitive radio networks depends on the primary user appearance. Recently, a spectrum-aware data replication approach has been proposed for intermittently connected cognitive radio networks [15]. However, it does not consider the cost for disseminating data to the caching/replicating nodes.

Due to primary user appearance, the delay in cognitive radio networks is much longer compared with traditional wireless networks, and it has been studied by several prior works. The distribution of transmission delay was first studied by Wang and Akyildiz [16]. They found that if the occupying time of licensed channels follows heavy tailed distribution, the transmission delay also follows heavy tailed distribution. However, they did not propose any solution to reduce the effect of transmission delay on network performance. Some channel assignment protocols have been proposed to minimize the total expected delay [17] or maximize network-wide throughput [18], but the performance may frequently change due to primary user appearance. To address this problem, some other protocols [19], [20] have been proposed to find the path that is least affected by primary users. However, none of the existing work can statistically bound the data access delay, which is the focus of this paper.

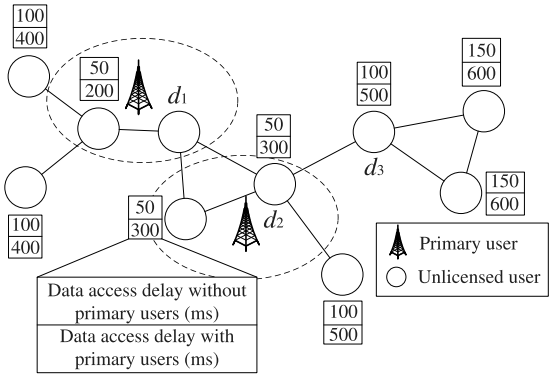


Fig. 1. Delay-constrained caching for data item d_1. The data access delay is at 90 percent confidence interval, i.e., the actual data access delay has 90 percent probability to be less than this value.

3 Overview

In this paper, we consider a caching scenario of multi-hop cognitive radio network consisting of unlicensed users (nodes) whose links may be affected by primary users. For example, in disaster recovery, a multi-hop cognitive radio network may consist of several rescue centers and a group of rescuers (or users). The rescue centers usually have powerful data centers, and the rescuers need to access the data centers to obtain various data such as evacuation plan, building blueprint or updated road condition, instructions or videos to help wounded person, etc. In this scenario, an ad-hoc network may have to be deployed since the cellular network infrastructure may be damaged. With cognitive radio techniques, more spectrum can be used, which makes the communications more robust in this challenging environment. The nearby rescuers (users) tend to have similar missions and hence share common interests in the data items. That is, if one user has accessed a data item from the data center, it is likely that nearby users want to access the same data some time later. It will save a large amount of energy, bandwidth and time if later accesses to the same data are served by the nearby user who caches the data instead of the far away data center. Some data may be updated periodically, and thus the data centers need to disseminate the updated data to the caching nodes. Similar examples exist in battlefield and vehicular networks.

Fig. 1 shows the caching scenario for data item d_1. Note that different data items (e.g., d_1, d_2, and d_3) may be generated by different nodes. The data access delay is generally increased due to the primary user appearance which affects the data transmission of unlicensed users. To reduce the data access delay, data item d_1 should be cached at appropriate nodes. In this paper, we consider that the total cost consists of two parts, i.e., the *dissemination cost* for the data source (e.g., the aforementioned data center) to disseminate the data to all caching nodes and the *access cost* for the requesters to access the data. The formal definition is given in Section 5.

The uncertain behavior of the primary users makes it challenging to correctly estimate the total cost. We propose

a model to formulate the behavior of primary users as a continuous-time Markov chain, based on which we derive the distribution of link transmission delay. Based on this model, we propose various techniques for determining caching locations, such that the delay constraint is satisfied for each node and the total cost is minimized. More specifically, we first propose a cost-based approach to minimize the total cost without considering the delay constraint. Then, we propose a delay-based approach which greedily caches data at the node that violates the delay constraint the most. Both approaches determine the caching nodes in a centralized manner. Thus, we propose a distributed hybrid approach to minimize the total cost subject to the delay constraint. The total cost is further reduced via local coordination among nodes themselves.

4 System Model

We consider a multi-hop cognitive radio network, where each unlicensed user has one cognitive radio to opportunistically access $\mathcal{C}$ licensed channels. The network is modeled as a connected graph $G(V, E)$ where each node $v \in V$ corresponds to an unlicensed user, and an edge $e = (u, v) \in E$ represents the link between u and v if they are within the transmission range. Each link e can work on a channel which is not currently accessed by primary users. Following several prior works [8], [9], the primary user appearance is modeled as the following continuous-time Markov chain, based on which the link transmission delay is probabilistically estimated.

For link e, we denote the current primary user appearance by $\mathbf{M}_e(t) = (M_{e,1}(t), M_{e,2}(t), \ldots, M_{e,\mathcal{C}}(t))$ where $M_{e,c}(t) = 1$ if channel c is accessed by some primary user; otherwise, $M_{e,c}(t) = 0$. $M_{e,c}(t)$ follows a continuous-time Markov chain with two states, state 1 ($M_{e,c}(t) = 1$) and state 0 ($M_{e,c}(t) = 0$). $q_{e,c}^{i,j}$ is the transition rate from state i to state j, and corresponds to the (i, j)th element of the generator matrix for Markov chain $M_{e,c}(t)$.

We assume that the primary user appearance on different channels is independent. Then, the composition of the corresponding $\mathcal{C}$ continuous-time Markov chains is still a continuous-time Markov chain. Let $Q_{e,c}$ be the generator matrix for Markov chain $M_{e,c}(t)$, then $\oplus_{c=1}^{\mathcal{C}} Q_{e,c}$ is the generator matrix for Markov chain $\mathbf{M}_e(t)$ [21]. Here $\oplus$ represents the *Kronecker sum* which is defined as follows.

Definition 1 (Kronecker Sum). *Let A be an $n \times n$ matrix, B be a $p \times p$ matrix, I_m be an $m \times m$ identity matrix. $A \oplus B = A \otimes I_p + I_n \otimes B$.*

The operation $\otimes$ in Definition 1 denotes Kronecker product, which is defined in Definition 2:

Definition 2 (Kronecker Product). *Let A be a $n \times m$ matrix, B be a $p \times q$ matrix, and C be a $np \times mq$ matrix. Let X_{ij} be the (i, j)th element of matrix X. $C = A \otimes B$ denotes that $C_{ij} = A_{i_1 j_1} B_{i_2 j_2}$ where $i = (i_1 - 1)p + i_2$ and $j = (j_1 - 1)q + j_2$.*

By defining a $\mathcal{C} \times 1$ vector $\mathbf{1} = (1, 1, \ldots, 1)'$, we use T_e^l to denote the time period that $\mathbf{M}_e(t)$ stays at state $\mathbf{1}$, which is equivalent to the time that link e waits until some licensed channel becomes available. Since $\mathbf{M}_e(t)$ is a continuous-time Markov chain, T_e^l is exponentially distributed with parameter $-q_e^{11}$ [22]. Here q_e^{11} is the element at row $\mathbf{1}$ and column $\mathbf{1}$ of the generator matrix for $\mathbf{M}_e(t)$.

Since our focus is to deal with the disruption caused by primary user appearance, for the transmission delay of link e, we will focus on the time waiting for available licensed channels (T_e^l) and using traditional techniques to deal with the delay caused by the wireless interference. That is, while considering the interference from primary user appearance, the wireless interference is simplified in order to facilitate the analysis on deriving the distribution of data access delay and checking the delay constraint.

5 Problem Formulation

We first formally define the cost (the dissemination cost and the access cost) and the delay constraint, and then formulate the delay-constrained caching problem. We assume the storage cost is insignificant because wireless devices are recently equipped with huge storage capacity. As a result, cache placement of different data items can be decoupled. To simplify the presentation, we will focus on cache placement of a specific data item in the rest of the paper.

5.1 Delay Constraint

Let α be the confidence level, and β be the delay threshold. For node v, the delay constraint is defined as $P(a_v \leq \beta) \geq \alpha$, where a_v is the data access delay of node v.

5.2 Cost

5.2.1 Dissemination Cost

We define the dissemination cost as the amount of bandwidth consumed for data dissemination.

Let $\mathcal{V}$ be the set of caching nodes (including the data source). The data is disseminated from the data source along a tree connecting all nodes in $\mathcal{V}$. Let $\mathcal{G}$ be such dissemination graph, and b be the traffic rate for data dissemination, which is defined as the data size divided by the data update interval. Let $L(\mathcal{G})$ be the total number of edges of $\mathcal{G}$, then the dissemination cost can be represented by $bL(\mathcal{G})$.

In our work, the data source builds a minimum Steiner tree connecting all caching nodes, and use it as the dissemination graph. This minimizes the dissemination cost since the minimum Steiner tree has the minimum total length (which can be the total number of edges if each edge has unit weight). Since the minimum Steiner tree problem is NP-hard, an approximation algorithm in [23] is used to build it. The data source periodically disseminates the updated data to the caching nodes along this dissemination graph. By organizing the caching nodes as a tree, each caching node disseminates the updated data to its children in a distributed and hierarchical manner.

5.2.2 Access Cost

We define the access cost as the aggregated data access delay, following several prior caching works [3], [13]. However, the data access delay may vary significantly from time to time due to the primary user appearance. This makes it difficult to define the access cost which should stay stable over a long period of time. To address this challenge, we

TABLE 1
Table of Notations

Delay Constraint:	
a_v	node v's access delay
α	confidence level
β	delay threshold
Dissemination Cost:	
b	traffic rate for data dissemination
$\mathcal{L}(\mathcal{G})$	the number of links used for data dissemination
Access Cost:	
p_v	node v's access probability
d_{uv}	transmission delay between u and v
$\mathcal{W}$	cost ratio

define the access cost as follows. Let d_{uv} be the transmission delay between nodes u and v. Then, $E(d_{uv})$ is the expected transmission delay between nodes u and v, which can be modeled by the length of the shortest u-v path in a weighted graph where the link weight is the mean link transmission delay. The access cost can then be represented by $\sum_v 2\min_{u\in\mathcal{V}} d_{vu}p_v$, where $2\min_{u\in\mathcal{V}} d_{vu}$ is the expected round-trip transmission delay from v to the nearest caching node, and p_v is the data access probability of node v (the probability that node v accesses the data in a period of time).

Generally speaking, as the number of caching nodes increases, the access cost decreases while the dissemination cost increases. Since wireless devices are usually equipped with limited amount of energy and bandwidth, the dissemination cost should not be too high, and the caching approach should balance the access cost and the dissemination cost. Since the access cost and the dissemination cost have different units, they cannot be simply added together. We use some cost ratio $\mathcal{W}$ to match the units and adjust the weight of the dissemination cost, in order to reflect the relative importance of the dissemination cost and the access cost. The effect of $\mathcal{W}$ on the network performance is further evaluated through simulations. The total cost is defined as the sum of the access cost and $\mathcal{W}$ times of the dissemination cost.

5.3 Delay-Constrained Caching Problem

Given notations in Table 1, the cache placement problem in cognitive radio networks can be formulated as the following delay-constrained caching problem

$$\textbf{minimize } \sum_v 2\min_{u\in\mathcal{V}} d_{vu}p_v + \mathcal{W}bL(\mathcal{G}),$$
$$\textbf{subject to } P(a_v \le \beta) \ge \alpha, \forall v.$$

The delay-constrained caching problem is NP-hard. If β is very large, the delay constraint is always satisfied. This reduces the problem to the same connected facility location problem addressed by *poach* [4] which is NP-hard in general.

Our problem is essentially a chance-constrained programming problem since the data access delay is a random variable. The chance-constrained programming problem is considered as very difficult and intractable [24]. There is even no general solution for chance-constrained programming problems. However, the structure of the problem may be exploited to design efficient solutions. Next we propose three caching approaches specific to our problem.

6 Cost-Based Approach

In this section, we first describe a linear programming formulation, and then present a primal-dual algorithm to minimize the total cost.

6.1 Linear Programming Formulation

In this approach, we formulate the problem as the following linear programming problem to minimize the total cost. Here v_0 is the data source, and p_u is the data access probability of node u. x_{uv} takes 1 or 0. $x_{uv} = 1$ if node u gets the data from node v which caches the data; otherwise, $x_{uv} = 0$. z_e is a binary variable indicating whether e is in dissemination graph $\mathcal{G}$ (defined in Section 5). $z_e = 1$ if e is in $\mathcal{G}$; otherwise, $z_e = 0$. $\delta(S)$ is the set of edges which have only one end-node belonging to S. d_{uv} is the expected transmission delay between nodes u and v (defined in Section 5). The objective function (1) is the total cost which is to be minimized. Constraint (2) ensures that each node gets the data from one caching node. Constraint (3) ensures that if node u gets the data from some caching node v, v is connected with v_0 in the dissemination graph. Equivalently, for any node set $S \subseteq V \setminus \{v_0\}$ which contains v, there must be at least some edges which have one end-node belonging to S and are also in the dissemination graph. Relaxing the integrity constraint (4) to $0 \le x_{uv} \le 1, 0 \le z_e \le 1$ gives us a linear program. Note that the delay constraint cannot be contained in this linear program, since a linear program cannot contain any random variable such as the data access delay. We will evaluate whether the cost-based approach satisfies the delay constraint or not through simulations

$$\textbf{minimize} \sum_{u\in V} p_u \sum_{v\in V} d_{uv}x_{uv} + \mathcal{W}b\sum_{e\in E} z_e \tag{1}$$

$$\textbf{subject to} \sum_{v\in V} x_{uv} = 1, \forall u \in V \tag{2}$$

$$\sum_{v\in S} x_{uv} \le \sum_{e\in\delta(S)} z_e, \forall S \subseteq V \setminus \{v_0\}, \forall u \in V. \tag{3}$$

$$x_{uv}, z_e \in \{0, 1\} \tag{4}$$

The dual of this linear program is as follows:

$$\textbf{maximize} \sum_{u\in V} \alpha_u \tag{5}$$

subject to

$$\alpha_u \le p_u d_{uv} + \sum_{S: v\in S, v_0\notin S} \theta_{S,u}, \forall u \in V, v \in V \setminus v_0 \tag{6}$$

$$\sum_{u\in V} \sum_{S: e\in\delta(S), v_0\notin S} \theta_{S,u} \le \mathcal{W}b, \forall e \in E \tag{7}$$

$$\alpha_u \le p_u d_{uv_0}, \forall u \in V \tag{8}$$

$$\alpha_u, \theta_{S,u} \ge 0. \tag{9}$$

Now we describe the logic behind the formulation of the dual problem. Each node u is associated with a price α_u to

pay for data access. The objective function (5) is the total network profit which is to be maximized. The payment goes towards the access cost and the dissemination cost. If u gets data from a caching node v (not the data source v_0), it is willing to pay at most the access cost $p_u d_{uv_0}$ and the dissemination cost $\sum_{S:v\in S,v_0\notin S}\theta_{S,u}$ as shown in Constraint (6) . Here $\theta_{S,u}$ denotes u's payment for compensating the data source to disseminate data to the node set S (which the caching node v belongs to). Constraint (7) ensures that such type of payment goes towards the dissemination cost along each edge (i.e., $\mathcal{W}b$). If u gets data from v_0 directly, it only incurs the access cost as shown in Constraint (8) . Compared with the primal problem, the optimal value of the objective function (5) is at most the optimal value of the objective function (1) .

Our formulated problem is a special case of the connected facility location problem, in which we are given the location of an existing facility along with a set of locations at which further facilities can be built. Each location has demands which must be served by one facility, and all facilities must be connected by a Steiner tree. The objective is to find a solution to minimize the sum of serving cost and connection cost. In our problem, each node represents a location, the data source represents the existing facility, and caching nodes represent new facilities. Thus, we transform an existing primal-dual algorithm [12] for the connected facility location problem to address our problem.

6.2 Algorithm Description

The general idea of our primal-dual algorithm is to find a solution for the dual problem, based on which a solution for the primal problem can be constructed. We summarize the overall flow in Algorithm 1 and describe the detailed algorithm as follows.

Algorithm 1. Cost-Based Approach

Input: graph $G(V, E)$, p_u, d_{uv} for $\forall u, v \in V$
Output: x_{uv} for $\forall u, v \in V$, and z_e for $\forall e \in E$

1: Construct a dual problem in which α_u and $\theta_{S,u}$ are initialized to 0 for all u, S
2: Raise α_u and $\theta_{S,u}$ in the aforementioned controlled manner to select candidate caching nodes
3: Find a maximal independent set $\mathcal{V}$ from all candidate caching nodes
4: Assign each node u to access data from some caching node v, i.e., $x_{uv} \leftarrow 1$
5: Build a dissemination tree to connect all nodes in $\mathcal{V}$, i.e., $z_e \leftarrow 1$ if e is in the dissemination tree

6.2.1 Finding a Solution for the Dual Problem

Suppose there is a notion of time t. Initially $t = 0$, $\alpha_u = \theta_{S,u} = 0$ for all u, S, and the data source v_0 is selected as a candidate caching node. As time increases, α_u is raised at unit rate, i.e., $\alpha_u = t$. At some time, α_u is equal to $p_u d_{uv}$ for some node v, which is called u becomes *tight* with node v.

Let $\mathcal{S}_u$ denote the set of nodes with which u is tight. After u becomes tight with another non-caching node, both α_u and $\theta_{\mathcal{S}_u,u}$ will be raised at unit rate until one of the following events happens.

- There is a non-caching node v on some edge (v_1, v_2) with which $d_{v_1v_2}\sum_{u'\in\mathcal{T}_v} p_{u'} \geq \mathcal{W}b$ where $\mathcal{T}_v$ denotes the set of nodes tight with v. Then, we select v as a candidate caching node and stop raising α_u for all nodes in $\mathcal{T}_v$.[1]
- u becomes tight with a candidate caching node.

The aforementioned process continues until every node is tight with a candidate caching node. Lemma 1 demonstrates that all constraints in the dual problem are satisfied during the aforementioned process, which guarantees the correctness of our primal-dual algorithm.

Lemma 1. *α_u and $\theta_{S,u}$ satisfy all constraints in the dual problem during the aforementioned process.*

Proof. We first show Constraint (6) and (8) are satisfied. Initially, $\alpha_u = \theta_{S,u} = 0$, so the left hand side (i.e., 0) is less than the right hand side (i.e., $p_u d_{uv}$) which is positive. These inequalities hold until u becomes tight with v. If v is the data source v_0, α_u will not be raised according to the aforementioned process, which makes Constraint (8) still satisfied. If v is a non-caching node, α_u and $\sum_{S:v\in S,v_0\notin S}\theta_{S,u}$ are raised at the same rate, which makes Constraint (6) still satisfied.

Now we show Constraint (7) is satisfied. Consider an edge $e = (v_1, v_2)$. Without loss of generality, suppose $d_{uv_1} \leq d_{uv_2}$ for a node u. Thus, u becomes tight with v_1 first. After this happens (i.e., $\alpha_u = p_u d_{uv_1}$), α_u and $\sum_{S:v\in S,v_0\notin S}\theta_{S,u}$ are raised at the same rate, which makes u tight with increasingly more points on edge e. For each point v on this edge, define $f(u, v)$ to be p_u if u is tight with v, and 0 otherwise. We have $\sum_{S:e\in\delta(S),v_0\notin S}\theta_{S,u} = \int_{v_1}^{v_2} f(u, v)dv$, and hence $\sum_{u\in V}\sum_{S:e\in\delta(S),v_0\notin S}\theta_{S,u} = \sum_{u\in V}\int_{v_1}^{v_2} f(u, v)dv$. Since $\sum_{u\in V}\int_{v_1}^{v_2} f(u, v)dv = \int_{v_1}^{v_2}(\sum_{u\in V} f(u, v))dv = \int_{v_1}^{v_2}(\sum_{u\in\mathcal{T}_v} p_u)dv$ and the inequality $d_{v_1v_2}\sum_{u\in\mathcal{T}_v} p_u \leq \mathcal{W}b$ holds during the aforementioned process, we have $\int_{v_1}^{v_2}(\sum_{u\in V} f(u, v))dv \leq \mathcal{W}b$ which indicates Constraint (7) is satisfied. □

6.2.2 Constructing a Solution for the Primal Problem

Now we select caching nodes from the candidates. Let $\mathcal{V}'$ be the set of candidate caching nodes. We say that two candidate caching nodes v_1, v_2 are *dependent* if there is a node which is tight with both v_1 and v_2. A set of candidate caching nodes is said to be *independent* if no two nodes in this set are dependent. We want to find a maximal independent set $\mathcal{V}$ of $\mathcal{V}'$ such that $\mathcal{V}$ is independent and is not a subset of any other independent set. By selecting the nodes in $\mathcal{V}$ as caching nodes, the number of caching nodes can be reduced while ensuring every node can access data from at least one caching node. The detailed procedure for finding $\mathcal{V}$ is as follows:

1) Initialize $\mathcal{V}$ to be empty.
2) Sort the candidate caching nodes in $\mathcal{V}'$ in their order of becoming candidates.

1. Here v can also be any non-vertex node on some edge (v_1, v_2). This case will be handled by the technique developed in an earlier work on the connected facility location problem [12].

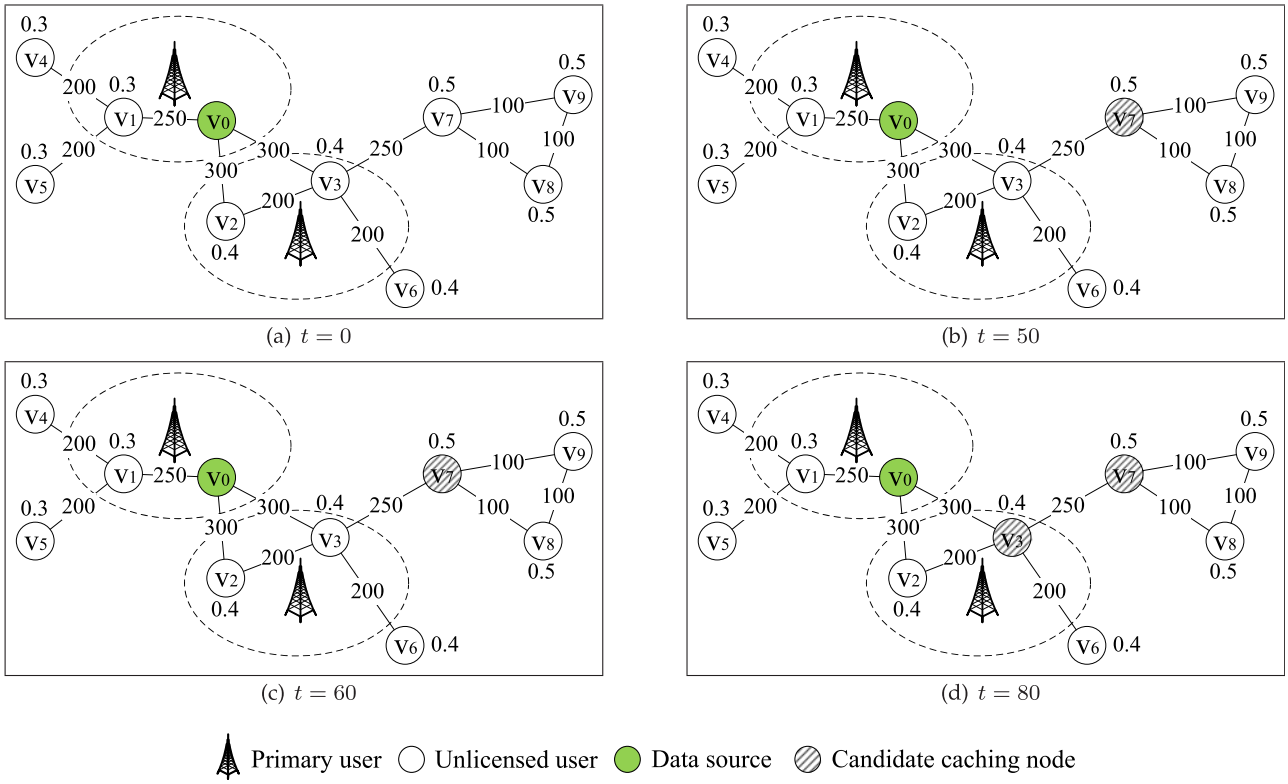


Fig. 2. Illustration of our primal-dual algorithm.

3) Considering nodes in $\mathcal{V}'$ in the sorted order, add a node to $\mathcal{V}$ if it is not dependent with any node already in $\mathcal{V}$.

Then, we assign a node u to access data from some caching node v (i.e., set $x_{uv} = 1$) as follows. If u is tight with some caching node $v \in \mathcal{V}$, assign u to v. Otherwise, let v' be the candidate caching node (in $\mathcal{V}'$) with which u is tight. According to the above procedure, there must be some caching node $v \in \mathcal{V}$ such that v and v' are dependent. We will assign u to v.

We build a dissemination tree on $\mathcal{V}$ (i.e., set z_e) as follows. Suppose each edge has unit weight. We first augment graph G to include all caching nodes in $\mathcal{V}$, and then find a *minimum spanning tree* to connect all these nodes. The weight of such minimum spanning tree is at most twice the optimal weight [23].

Theorem 1. *Our algorithm is guaranteed to generate a solution whose total cost is at most $3 + 2\mathcal{W}$ times that of the optimal solution.*

Proof. Let OPT be the total cost of optimal solution, and $\tilde{\alpha}_u$ be the value of α_u at the end of the aforementioned process. Following Lemma 3.2 in the previous paper on the facility location problem [12], node u's access cost $p_u d_{uv}$ is at most $3\tilde{\alpha}_u$ assuming v is the caching node assigned to u. Thus, the accumulate access cost of all nodes in the network is at most $3\sum_{u\in V}\tilde{\alpha}_u$. According to the duality theorem, the value of dual objective $\sum_{u\in V}\tilde{\alpha}_u$ is at most the value of primal objective which is the total cost to be minimized. Thus, the accumulate access cost is at most $3OPT$. Since the weight of our dissemination tree is at most twice the optimal weight, the dissemination cost is at most $2OPT$. Thus, the total cost of our algorithm is at most $(3 + 2\mathcal{W})OPT$. □

Theorem 2. *The time complexity of the cost-based approach is $O(m \log\ m)$ where m is the number of links.*

Proof. Following Theorem 8 in the paper on the facility location problem [25], finding a solution for the dual problem takes $O(m \log\ m)$ time. The running time in constructing a solution for the primal problem is dominated by solving the minimum spanning tree problem, whose time complexity is $O(m \log n)$ (n is the number of nodes) [26]. Thus, the overall time complexity is $O(m \log n)$ ($n < m$ in a connected graph). □

We use an example shown in Fig. 2 to illustrate our primal-dual algorithm. The network has 10 nodes in which v_0 is the data source. The label associated with a node (v) indicates the data access probability of that node (p_v), and the label associated with a link ((u, v)) indicates the expected transmission delay of that link (d_{uv}). Suppose $\mathcal{W}b = 200$.

Initially $t = 0$ (Fig. 2a), α_v is set to zero for each node v. When t increases to 50 (Fig. 2b), α_v will be 50 for each node v. This causes node v_9 to be tight with node v_7 ($\alpha_{v_9} \geq p_{v_9} d_{v_7 v_9}$). Similarly, it can be shown that node v_8 is also tight with node v_7, and that node v_7 is tight with itself. Let $\mathcal{T}_{v_7}$ denote the set of nodes tight with node v_7. Then, $d_{v_3 v_7}\sum_{v\in\mathcal{T}_{v_7}} p_v = 250 \times (0.5 + 0.5 + 0.5) = 375$, which is larger than $\mathcal{W}b$ (200). According to our primal-dual algorithm, node v_7 becomes a candidate caching node, and v_8, v_9 can access data from v_7. Following similar procedure, v_1, v_3 become candidate caching nodes at $t = 60$ (Fig. 2c) and $t = 80$ (Fig. 2d), respectively.

As every node is tight with a candidate caching node, the primal problem has a solution. All candidates v_1, v_3, v_7 are selected as caching nodes since there is no node tight with

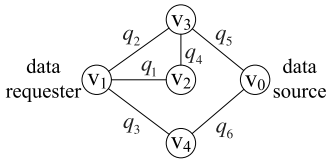


Fig. 3. An example of a data source. Each edge is associated with the transmission delay represented by an exponential distribution. The label associated with each link represents the parameter of the distribution.

any two of them. We assign v_4, v_5 to (access data from) v_1, and v_2, v_6 to v_3, and v_8, v_9 to v_7. The path v_1-v_0-v_3-v_7 will be used to disseminate data from the data source to all the caching nodes.

7 Delay-Based Approach

In this section, we first describe how to derive the distribution of data access delay, and then discuss how to check the delay constraint. At the end of this section, we present the delay-based cache placement algorithm for satisfying the delay constraint.

7.1 Distribution of Data Access Delay

The problem of deriving the distribution of data access delay can be reduced to the following problem. In the network graph, suppose each edge has an edge length equal to the corresponding link transmission delay, which is an exponentially distributed random variable. Then, the length of the shortest u-v path is equal to the transmission delay between u and v, which is denoted by d_{uv}. If the query is flooded over the network from u to all caching nodes (this assumption is relaxed in the hybrid approach), u's data access delay is approximately twice of $\min_{v\in\mathcal{V}} d_{uv}$. Here $\min_{v\in\mathcal{V}} d_{uv}$ is the minimum of all d_{uv}'s in which v is a caching node (the data source is regarded as a caching node). If we obtain the distribution of $\min_{v\in\mathcal{V}} d_{uv}$, the distribution of u's data access delay is obtained.

In the literature, Kulkarni has proposed an algorithm [27] to derive the distribution of the shortest path length from one node to another node, in a graph where the edge length is exponentially distributed. That algorithm cannot be directly applied since our problem considers the shortest path length from a data requester to multiple caching nodes. Next, we describe how to extend it to our problem.

7.1.1 Distribution of the Shortest Path Length from a Data Requester to a Data Source

Suppose a query is broadcast from the requester at time 0. The query is flooded over the network and travels at unit speed so that the time for traversing each link is equal to the corresponding edge length. Thus, the time when the data source first receives the query is equal to the shortest path length.

For example, in Fig. 3, suppose a query is broadcast from v_1 at time 0 when nodes v_1-v_2 and nodes v_1-v_4 do not share any common channel due to primary user appearance. Since nodes v_2 and v_4 have to wait some time before receiving the query, the query first reaches node v_3 before it reaches nodes v_2 and v_4. Even if node v_2 receives the query some time later, it can only send the query to either node v_1 or node v_3 which have already received the query. Thus, node v_2 is useless for the propagation of the query towards the data source (node v_0).

TABLE 2
State Space for the Example Network

State	State Description
1	$\{v_1\}$
2	$\{v_1,v_2\}$
3	$\{v_1,v_4\}$
4	$\{v_1,v_2,v_4\}$
5	$\{v_1,v_2,v_3\}$
6	$\{v_1,v_2,v_3,v_4\}$
7	$\{v_1,v_2,v_3,v_4,v_0\}$

Let $\mathcal{N}(t)$ be the set of nodes which have already received the query or are useless for the propagation of the query towards the data source at time t. According to [27], $\{\mathcal{N}(t), t \geq 0\}$ is a continuous-time Markov chain. The state space for the example graph (Fig. 3) is shown in Table 2, and the generator matrix Q is shown in Table 3. Let Ω_i be the ith state, and s be the number of states. In this example, $s = 7$. If the state is the final state Ω_s, the data source (node v_0) receives the message, and ends the aforementioned process. The time for $\mathcal{N}(t)$ to make transition from the initial state Ω_1 to the final state Ω_s is the shortest path length from the requester v_1 to the data source v_0, which is also equal to the transmission delay between v_1 and v_0 (denoted by $d_{v_1v_0}$). We have

$$d_{v_1v_0} = \min\{t \geq 0 : \mathcal{N}(t) = \Omega_s | \mathcal{N}(0) = \Omega_1\}.$$

Let $F(t)$ be the cumulative distribution function of $d_{v_1v_0}$, i.e., $F(t) = P(d_{v_1v_0} \leq t)$. $F(t)$ has no closed-form expression, and can be approximated by $F_k(t)$ and $\tilde{F}_k(t)$ defined as follows:

$$F_k(t) = 1 - \sum_{i=0}^{k} (1-\lambda_i) e^{-qt} (qt)^i / i!$$

$$\tilde{F}_k(t) = \lambda_k - \sum_{i=0}^{k} (\lambda_k - \lambda_i) e^{-qt} (qt)^i / i!$$

with

$$F_k(t) \leq F(t) \leq \tilde{F}_k(t)$$
$$0 \leq \tilde{F}_k(t) - F_k(t) \leq 1 - \lambda_k,$$

where λ_k is the $(1,n)$th element in the kth power of $Q^* = [q^*_{ij}]$. Here $q^*_{ij} = \delta_{ij} + q_{ij}/q$, where q_{ij} is the (i,j)th element in the generator matrix, $q = \max_{1\leq i\leq n}\{-q_{ii}\}$, and δ_{ij} is an indicator function, i.e., $\delta_{ij} = 1$ if $i = j$; otherwise, $\delta_{ij} = 0$.

Property of λ_k. $0 \leq \lambda_k \leq 1$ *and* λ_k *increases to* 1 *as* $k \to \infty$.

7.1.2 Distribution of the Shortest Path Length from a Data Requester to Multiple Caching Nodes

We use an example graph (Fig. 4) to show how to derive the distribution of the shortest path length from a data requester to multiple caching nodes. If node v_4 also caches the data, the aforementioned process should be adjusted as follows. That is, if either node v_4 or node v_0 receives the

TABLE 3
Generator Matrix for the Example Network

Q_{ij}	1	2	3	4	5	6	7
1	$-(q_1+q_2+q_3)$	q_1	q_3	0	q_2	0	0
2	0	$-(q_2+q_3+q_4)$	0	q_3	q_2+q_4	0	0
3	0	0	$-(q_1+q_2+q_6)$	q_1	0	q_2	q_6
4	0	0	0	$-(q_2+q_4+q_6)$	0	q_2+q_4	q_6
5	0	0	0	0	$-(q_3+q_5)$	q_3	q_5
6	0	0	0	0	0	$-(q_5+q_6)$	q_5+q_6
7	0	0	0	0	0	0	0

message, the process can be ended and thus states 3, 4, 6, 7 of Table 2 are merged into one final state. Then the distribution function of the shortest path length can be calculated accordingly (see [28] for the updated state space and the corresponding generator matrix).

7.2 Checking the Delay Constraint

Let α be the confidence level, and β be the delay threshold. Suppose we want to check the delay constraint for some data requester v. The data access delay (denoted by a_v) is twice of the shortest path length from the requester to the caching nodes. Recall that the cumulative distribution function of such shortest path length is denoted by $F(t)$ and the delay constraint is $P(a_v \leq \beta) \geq \alpha$. We can transform the delay constraint as follows:

$$F(\beta/2) \geq \alpha.$$

In our previous work [28], the approximation $F_k(t)$ is used to check the delay constraint since it is impossible to get the exact value of $F(t)$. Here k should be large enough to make $F_k(t) - \tilde{F}_k(t) < \epsilon$ (ϵ is a very small positive value) so that $F_k(t)$ is a good approximation. However, it is unnecessary to obtain the exact value of k. Thus, we can calculate $F_i(t)$, $\tilde{F}_i(t)$ $(i = 0, 1, \ldots, k)$ in an iterative manner and check the delay constraint based on the relationship $F_k(t) \leq F(t) \leq \tilde{F}_k(t)$ at each iteration. Once a conclusion can be reached at some iteration, the process can be terminated. The details are as follows.

We define $f_k(t) = \sum_{i=0}^{k} e^{-qt}(qt)^i/i!$. Then, $F_k(t)$, $\tilde{F}_k(t)$ and $f_k(t)$ can be incrementally calculated from $F_{k-1}(t)$, $\tilde{F}_{k-1}(t)$ and $f_{k-1}(t)$, by using Equations (10)-(12) . Note that we also need to calculate the kth power of Q^* for obtaining λ_k. The kth power of Q^* can be incrementally calculated from the $k-1$th power Q^*, in order to make the calculation of λ_k more efficient

$$F_k(t) = F_{k-1}(t) - (1-\lambda_k)e^{-qt}(qt)^k/k! \tag{10}$$

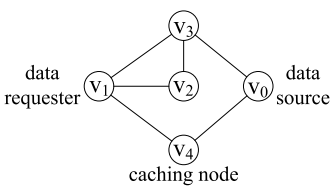


Fig. 4. An example of multiple caching nodes, where each edge is associated with the transmission delay represented by an exponential distribution.

$$\tilde{F}_k(t) = \tilde{F}_{k-1}(t) + (\lambda_k - \lambda_{k-1})(1 - f_{k-1}(t)) \tag{11}$$

$$f_k(t) = f_{k-1}(t) + e^{-qt}(qt)^k/k! \tag{12}$$

To check whether $F(\beta/2) \geq \alpha$ (the delay constraint) is satisfied, we first calculate $F_0(\beta/2)$ and $\tilde{F}_0(\beta/2)$. Since $F_k(t) \leq F(t) \leq \tilde{F}_k(t)$, if $F_0(\beta/2) \geq \alpha$ is satisfied, then $F(\beta/2) \geq \alpha$, i.e., the delay constraint is satisfied; or if $\tilde{F}_0(\beta/2) < \alpha$ is satisfied, then $F(\beta/2) < \alpha$, i.e., the delay constraint is unsatisfied. Otherwise, we cannot decide the relationship between $F(\beta/2)$ and α, so we have to calculate $F_1(\beta/2)$ and $\tilde{F}_1(\beta/2)$ to check it again. The aforementioned process is repeated, until at some step we can decide whether $F(\beta/2) \geq \alpha$.

If the stopping criteria $1 - \lambda_k \leq \epsilon$ is satisfied, we can also end the checking process. Here ϵ is a very small positive value. Since $0 \leq \tilde{F}_k(t) - F_k(t) \leq 1 - \lambda_k$, the stopping criteria indicates the approximations $F_k(\beta/2)$ and $\tilde{F}_k(\beta/2)$ are very close to $F(\beta/2)$. On the other hand, $F_k(\beta/2)$ and $\tilde{F}_k(\beta/2)$ are around α since we cannot decide whether the delay constraint is satisfied. Thus, $F(\beta/2)$ must be very close to α. In this case, we assume the delay constraint is satisfied.

7.3 Algorithm Description

In Algorithm 2, we first initialize the set of caching nodes ($\mathcal{V}$) to include only the data source, and the set of nodes not satisfying the delay constraint ($\mathcal{U}$) to include all requesters. We check whether the delay constraint is satisfied for each node. For nodes satisfying the delay constraint, we remove them from $\mathcal{U}$. If $\mathcal{U}$ is not empty, the node violating the delay constraint the most is selected and added to $\mathcal{V}$. Here we use $\alpha - F_v^{\mathcal{V}}(\beta/2)$ ($F_v^{\mathcal{V}}(t)$ is defined in the algorithm) to record how much node v violates the delay constraint. Since the exact value of $F_v^{\mathcal{V}}(\beta/2)$ cannot be obtained, we use its lower bound (recall $F_k(\beta/2)$ in Section 7.2) as an approximation. Let the selected node be v. If v caches the data, its data access delay is zero, and v is removed from $\mathcal{U}$. We check again whether the delay constraint is satisfied for all other nodes in $\mathcal{U}$. If not, the aforementioned procedure is repeated until $\mathcal{U}$ is empty. Otherwise, we have selected enough caching nodes.

Theorem 3. *The time complexity of delay-based approach is $O(n^2 s^{k-1})$ where n is the number of nodes, s is the number of states, and k is the parameter in $F_k(\beta/2)$.*

Proof. Consider the worst case that the data must be cached by all nodes in order to satisfy the delay constraint. According to Algorithm 2, there will be n iterations in

the while-loop. At the ith iteration, the delay constraint violation $\alpha - F_k(\beta/2)$ is calculated for $n-i+1$ nodes and the node violating the delay constraint the most is selected. In order to calculate $F_k(\beta/2)$, we must obtain λ_i $(i = 1, 2, \ldots, k)$ which is an element in the ith power of Q^* (see Section 7.1.1). The calculation of these elements take $O(s^{k-1})$ time in which s is the number of states (i.e., Q^* is an $s \times s$ matrix). Thus, the overall time complexity of Algorithm 2 is $O(n^2 s^{k-1})$. □

Algorithm 2. Delay-Based Approach

Input: graph $G(V, E)$, confidence level α, delay threshold β
Output: set of caching nodes $\mathcal{V}$

```
1: V ← v0 /*v0 is the data source*/
2: U ← V \ {v0}
3: while U is not empty do
4:    for each node v in U do
5:       Check whether the delay constraint is satisfied
6:       If satisfied, remove v from U; otherwise, record
         α − F_v^V(β/2)/*F_v^V(t) denotes F(t) (see Section 7.2),
         with v as a requester and V as the set of caching
         nodes*/
7:    end for
8:    Select the node v with the largest α − F_v^V(β/2) among all
      the nodes in U
9:    V ← V ⋃ {v}
10:   U ← U \ {v}
11: end while
12: Return V
```

8 Hybrid Approach

In this section, we describe the data structure maintained at each node for distributed cache placement and data access. Then, we discuss how to check the delay constraint in a distributed manner, and present the distributed algorithm.

8.1 Data Structure

8.1.1 Caching Node List

Each node maintains a subset of caching nodes in a node list, which is built and updated in the distributed algorithm.

8.1.2 Caching Path List

Each node selects a path to each node in the caching node list, and maintains these paths in a path list. Next we describe how to select these paths. Suppose node v wants to select a path towards node u which is in the caching node list. Similar to routing discovery in AODV [29], v discovers a number of paths to u, during which the delay information of each path (the distribution parameter of link transmission delay) is also collected. If a node collects more than one path to the caching node, the path with the minimum expected transmission delay is selected.

If a node v wants to access data, it multicasts queries along the pre-selected paths to all caching nodes in the caching node list. More specifically, node v uses source routing to send the query to caching node u. That is, the query includes a v-u path as a subfield, so other nodes in the network know to which node to forward the message. Through source routing, v can force the query to be forwarded along a path chosen by itself. By choosing a good path, the data access delay can be statistically bounded.

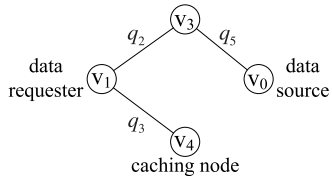


Fig. 5. The graph for calculating the data access delay of node v_1. Nodes v_4 and v_0 are in the caching node list.

8.2 Checking the Delay Constraint

Based on the caching node list and the caching path list, each node derives the distribution of data access delay, so that the delay constraint can be checked.

Let N_v be node v's caching node list, and let P_v^u be node v's caching path for caching node u. Node v constructs a graph $G_v(V_v, E_v)$. Here V_v includes v, all nodes in N_v, and all nodes on path P_v^u for $\forall u \in N_v$. E_v includes all edges on path P_v^u for $\forall u \in N_v$. For example, node v_1 constructs graph G_{v_1} as shown in Fig. 5. Here nodes v_4 and v_0 are in node 1's caching node list. The caching path for node v_4 is v_1-v_4, and the caching path for node v_5 is v_1-v_3-v_0.

If each edge has length equal to the corresponding link transmission delay, then the data access delay of node v is twice of the shortest path length from requester v to the caching nodes in N_v. The distribution of data access delay can be derived by applying aforementioned techniques in Section 7.1.2 to G_v.

8.3 Distributed Algorithm

The distributed algorithm consists of two procedures, *caching node selection* and *update*. The caching node selection procedure selects enough caching nodes so that the delay constraint is satisfied for all requesters. When some node becomes a caching node, it informs the data source, and the data source informs all requesters of the new caching node. Then each requester uses the update procedure to update the caching node list and the caching path list.

8.3.1 Caching Node Selection

The delay-based approach greedily caches data at the node that violates the delay constraint the most. To implement it in a distributed environment, each node informs all its neighbors of its average access delay and whether the delay constraint is satisfied. For a specific node v, if the delay constraint has not been satisfied by some of its neighbors with higher average access delay, it must wait until all the neighbors with higher average access delay satisfy the delay constraint; otherwise, node v requests to be a caching node as follows if the delay constraint is not satisfied at node v.

Each node broadcasts a request to all its neighbors, and decides to cache the data only if the request is accepted by all its neighbors. Let the node be v. For neighbor u, it accepts the request if (i) it is not requesting to be a caching node, (ii) it has not accepted requests from any other neighbor who requests to be a caching node at the same time. Otherwise, u rejects the request. Based on the acknowledgements (either

accept or reject), v decides whether to cache the data or not, and informs the neighbors of the decision, so that they may request to be a caching node or accept requests from other nodes later on. Note that if v's request got rejected, v will wait for a random amount of time and then re-check the condition for requesting to be a caching node.

Reducing the total cost. After the delay constraint is satisfied, the total cost may still be reduced by placing more caching nodes. For node v, it estimates the amount of reduction in total cost if it caches the data, i.e.,

$$\Delta_v = p_v E(a_v) - \mathcal{W} b h_v.$$

Here p_v is node v's access probability, $E(a_v)$ is v's average data access delay, b is the traffic rate for data dissemination (defined in Section 5), and h_v is the number of hops from node v to the nearest caching node in the caching node list. p_v and $E(a_v)$ can be measured by node v itself, b can be obtained from the data source, and h_v can be known from node v's caching node list. If $\Delta_v > 0$, v requests to be a caching node following the aforementioned procedure. Otherwise, nothing is done.

Theorem 4 demonstrates the correctness of the caching node selection procedure.

Theorem 4. *The caching node selection procedure terminates in finite steps; upon termination, the delay constraint is satisfied for all nodes.*

Proof. According to the procedure, if the delay constraint is not satisfied at some node v, v will cache the data. After the data is cached, the data access delay becomes zero, which indicates the delay constraint is satisfied. Thus, the delay constraint is satisfied for all nodes when the procedure terminates. Since there are finite number of nodes in the network, it takes finite steps to terminate the procedure. □

8.3.2 Update

Suppose node v is informed that u becomes a new caching node. Following the techniques aforementioned in Section 8.1.2, v obtains the v-u path with the minimum expected transmission delay. The update procedure is divided into two cases. If the caching node list N_v is empty, we add u to N_v, and assign the caching path P_v^u to be the v-u path with minimum expected transmission delay. Otherwise, we set an upper limit γ on the size of N_v. This is because if the caching node list has too many nodes, a huge amount of traffic would be generated by multi-casting a query towards these caching nodes.

(i) $|N_v| < \gamma$: node u is added to N_v^u, and the v-u path with the minimum expected transmission delay is assigned to P_v^u.

(ii) $|N_v| = \gamma$: node v decides whether u should replace some caching node in N_v and update the caching path accordingly. It mainly depends on whether such replacement has benefit. That is, it reduces the violation in the delay constraint (which can be calculated by $\alpha - F_v^{\mathcal{Y}}(\beta/2)$ as presented in Section 7.3).

First of all, for each node in N_v, we check whether replacing it with the new caching node u has the aforementioned benefit. If so, we add it to a set N_v' and records the amount of benefit in terms of the violation in the delay constraint. If N_v' is empty, v does not perform any replacement. Otherwise, it selects the node (denoted by u') which has the maximum benefit among all nodes in N_v'. Then, v replaces u' with u, removes $P_v^{u'}$ from the caching path list, and assign P_v^u to be the v-u path with the minimum expected transmission delay.

The hybrid approach is summarized by Algorithm 3.

Theorem 5. *The time complexity of hybrid approach is $O(n^2)$ where n is the number of nodes.*

Proof. In this proof, we assume requests are generated in order so that there is no contention between any two requests. Consider the worst case that all nodes are in the interference range of each other and that the data must be cached by all nodes in order to satisfy the delay constraint. According to Algorithm 3, the node with the highest average data access delay becomes the caching node. It takes $O(n)$ time to inform the data source and the other data requesters. Since all n nodes have to cache the data, the overall time complexity is $O(n^2)$. □

Algorithm 3. Hybrid Approach

Input for node v: graph $G(V, E)$, caching node list N_v, caching path P_v^u ($\forall u \in N_v$).

if the delay constraint is unsatisfied or $\Delta_v > 0$ **then**
  If some neighbor with higher average data access delay does not satisfy the delay constraint, return.
  Request to be a caching node and broadcast the request.
  If the request is accepted by all neighbors, cache the data and inform the data source.
  Inform the neighbors of the decision.
end if
if node u becomes a new caching node **then**
  if $|N_v| = \gamma$ **then**
    $N_v' \leftarrow \Phi$
    For each node in N_v, add it to N_v' if replacing the node with u has benefit.
    If N_v' is empty, return.
    $u' \leftarrow$ the node which has the maximum benefit among all nodes in N_v'.
    Remove u' from N_v, and remove the caching path list $P_v^{u'}$.
  end if
  Add u to N_v
  $P_v^u \leftarrow$ the v-u path with the minimum expected transmission delay (selected following Section 8.1).
end if

9 Performance Evaluations

In this section, we evaluate the performance of our approaches through extensive simulations.

9.1 Simulation Setup

In our simulations, we randomly place 25 nodes in a $3{,}200 \times 300\,\mathrm{m}^2$ area. The transmission range is $250\,\mathrm{m}$. If two nodes are within the transmission range, there is a link between them. There are 10 channels, which may be

sometimes accessed by primary users and become unavailable. For each channel, the unavailable period follows an exponential distribution whose mean value is between 1 and 10 s.

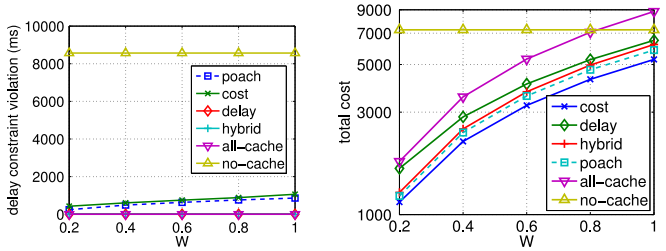
We use the same data query model as in previous studies [3], [6], [13]. Each node generates a stream of queries. The query generation time follows exponential distribution with mean value 5 s. After a query is sent out, the node does not generate new query until the query is served. The data access pattern is based on Zipf-like distribution, in which the access probability of the ith data item is proportional to $i^{-\theta}$. Here θ shows how skewed the access distribution is and we choose θ to be 0.8 based on studies on real web traces [30]. The access pattern is location-dependent; nodes around the same location tend to access similar data. The whole simulation area is divided into 10 (X-axis) by 2 (Y-axis) grids. These grids are named grid $0, 1, 2, \ldots 19$ in a column-wise fashion. For each node in grid i, if the generated query should access data id according to the original Zip-like access pattern, the new id would be $(id + n \bmod i) \bmod n$, where n is the database size ($n = 100$ in our simulation).

The data items are generated by two nodes, node 0 and node 24; data items with even id's are generated by node 0 and the rests are generated by node 24. The data size is uniformly distributed between 100 B and 7 KB. The data update interval follows exponential distribution with mean value 10s.

To check the delay constraint, we use confidence level $\alpha = 0.9$. The delay threshold β is different for different experiments. The cost ratio $\mathcal{W}$ is also adjusted to study its effect on performance.

We evaluate our three caching approaches, cost-based (*cost*), delay-based (*delay*) and hybrid (*hybrid*). In *delay* and *hybrid*, parameters ϵ and γ are set to 10^{-5} and 3, respectively. We also compare them with caching approaches designed for traditional wireless networks. Specifically, we implement an existing approach *poach* [4] which balances the access delay and the energy cost (including disseminating data to all caching nodes) without considering the primary user appearance. We also implement two naive approaches, *all-cache* and *no-cache*. In *all-cache*, data is cached at all nodes so every node can access the data locally; in *no-cache*, no node caches the data so every node has to access the data from the data source. Compared with the other approaches, *all-cache* leads to almost zero access cost but the highest dissemination cost, whereas *no-cache* leads to zero dissemination cost but the highest access cost.

The evaluation is based on several metrics: the amount of delay constraint violation, the total cost, the access cost and the dissemination cost. The amount of *delay constraint violation* shows how far the data access delay deviates from the delay threshold (β), and it is calculated as follows. For each node v, we measure the access delay for each query and calculate t_v such that 90 percent of the measured access delay (a_v) does not exceed it, i.e., $P(a_v \leq t_v) = 90\%$. The amount of *delay constraint violation* is defined as the largest $t_v - \beta$ among the nodes that do not satisfy the delay constraint. If all nodes satisfy the delay constraint, the amount of delay constraint violation is zero.



(a) delay constraint violation (b) total cost

Fig. 6. Effects of the cost ratio ($\beta = 800$ ms).

9.2 Effects of the Cost Ratio ($\mathcal{W}$)

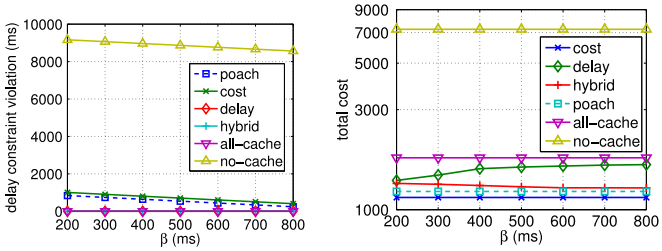
Fig. 6a shows the effect of $\mathcal{W}$ on the amount of delay constraint violation. *no-cache* has the highest amount of delay constraint violation 8,565 ms since all data has to be accessed from the data sources. For *poach* (*cost*), they are unaware of the delay constraint, and the amount of delay constraint violation increases as $\mathcal{W}$ increases. For example, when $\mathcal{W}$ increases from 0.2 to 1, the amount of delay constraint violation increases from 239 ms (404 ms) to 859 ms (1,011 ms). This is because increasing $\mathcal{W}$ increases the weight of the dissemination cost, and then these two cost-based approaches place less caching nodes to reduce the dissemination cost. This increases the data access delay and thus increases delay constraint violations. For *delay* and *hybrid*, the amount of delay constraint violation is zero since the delay constraint is satisfied for all nodes. For *all-cache*, the amount of delay constraint violation is also zero since all data can be accessed locally.

Fig. 6b shows the effect of $\mathcal{W}$ on the total cost. For *no-cache*, the total cost stays flat as $\mathcal{W}$ increases since the dissemination cost is zero (no data is cached). For the other approaches, the total cost increases as $\mathcal{W}$ increases since the dissemination cost is given more weight. *cost* has 6-10 percent lower total cost than *poach* since the link transmission delay is modeled more accurately by considering primary user appearance. Although *hybrid* also minimizes the total cost in the procedure of caching node selection, it has to satisfy the delay constraint at the same time, so it has 11-17 percent higher total cost than *cost*. *delay* does not minimize the total cost, and it has 5–29 percent higher total cost than *hybrid*. *all-cache* and *no-cache* have the highest total cost.

9.3 Effects of the Delay Threshold (β)

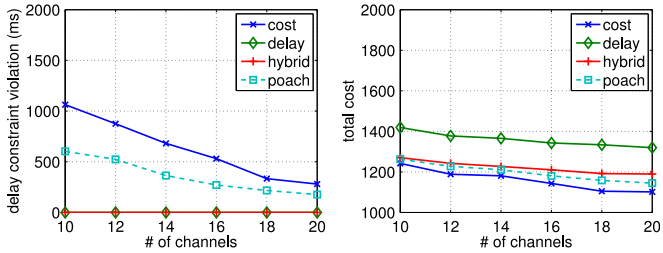
Fig. 7a shows the effect of β on the amount of delay constraint violation. For *poach* (*cost*, *no-cache*), the amount of delay constraint violation decreases as β increases; the amount of delay constraint violation decrease is actually the amount of increase of β. For example, when β increases from 200 to 800 ms, the amount of delay constraint violation decreases from 839 ms (1,004, 9,165 ms) to 239 ms (404, 8,565 ms). *no-cache* has the highest amount of delay constraint violation. For *delay*, *hybrid* and *all-cache*, the delay constraint is satisfied for all nodes, so the amount of delay constraint violation is zero.

Fig. 7b shows the effect of β on the total cost. For *poach*, *cost*, *all-cache*, and *no-cache*, the total cost stays flat as β increases. These four approaches are unaware of the delay constraint, so they are unaffected by the delay threshold β. As mentioned before, *cost* has the lowest total cost since



(a) delay constraint violation (b) total cost

Fig. 7. Effects of the delay threshold ($\mathcal{W} = 0.2$ ms/bps).



(a) delay constraint violation (b) total cost

Fig. 9. Effects of the number of channels.

minimizing the total cost is the main goal. The total cost of *cost* is 35 percent (84 percent) lower than that of *all-cache* (*no-cache*), and 6 percent lower than that of *poach* since the link transmission delay is modeled more accurately by considering primary user appearance. *all-cache* and *no-cache* have the highest total cost.

For *delay*, the total cost increases as β increases from 200 to 800 ms. Increasing β relaxes the delay constraint, so less nodes are needed to cache the data. Since *delay* does not minimize the total cost, this leads to 19 percent increase of the total cost. For *hybrid*, the total cost decreases by 5 percent as β increases from 200 to 800 ms. As the delay constraint is relaxed, *hybrid* focuses more on minimizing the total cost (in the procedure of caching node selection), and thus the total cost decreases. Among the approaches satisfying the delay constraint (*hybrid* and *delay*), *hybrid* has 3-29 percent lower total cost than *delay*.

We also evaluate the caching approaches in different experimental scenarios, such as different network topologies (as shown in Fig. 8), different number of channels (as shown in Fig. 9) and different data access pattern (as shown in Fig. 10). In all experiments, $\mathcal{W}$ and β are set to 0.2 ms/bps and 800 ms, respectively.

9.4 Effects of the Network Topology

We also evaluate all caching approaches under different network topologies which are generated as follows. Following the simulation setup in Section 9.1, 25 nodes are randomly placed in a $3{,}200 \times 300\text{m}^2$ area which is divided into 10 (X-axis) by 2 (Y-axis) grids. Here we also ensure that each grid contains at least one node. By increasing the transmission range from 250 to 500 m, each node is connected with more neighboring nodes, so we will obtain network topologies from a sparsely connected graph to a densely connected graph.

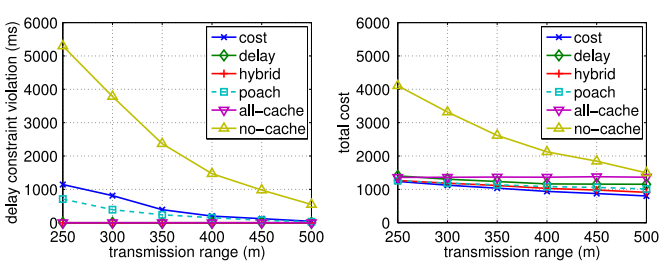
Fig. 8a shows the effect of network topology on the amount of delay constraint violation. For *poach*, *cost* and *no-cache*, the amount of delay constraint violation decreases as the transmission range increases. This is due to the decrease of access cost since the data can be accessed within fewer hops. The approaches *delay*, *hybrid* and *all-cache* satisfy the delay constraint at all times.

Fig. 8b shows the effect of network topology on the total cost. For *all-cache*, the total cost stays almost flat as the transmission range increases due to the following reason. Since data is cached everywhere in the network, the access cost is zero. For the dissemination cost, increasing the transmission range does not change the number of edges in the dissemination tree from the data source to all nodes, so the dissemination cost remains unchanged. For all other approaches, the total cost decreases as the transmission range increases. This is due to the decrease of access cost as mentioned earlier. Among the approaches satisfying the delay constraint (i.e., *hybrid*, *delay* and *all-cache*), *hybrid* has the lowest total cost, i.e, 10-21 percent lower than *delay* and 7-33 percent lower than *all-cache*.

9.5 Effects of the Number of Channels and the Zipf Parameter

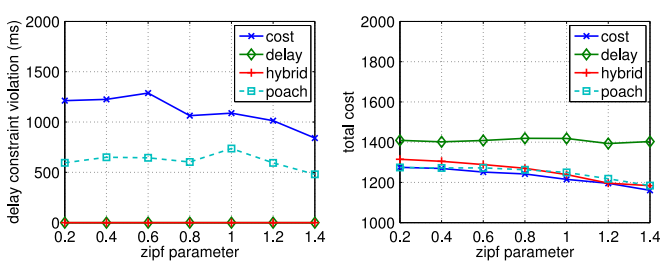
Fig. 9 shows the effects of the number of channels on the amount of delay constraint violation (Fig. 9a) and the total cost (Fig. 9b). In general, both the amount of delay constraint violation and the total cost decrease as the number of channels increases. This is because the access delay (cost) is reduced by increasing the number of channels, since it reduces the time waiting for an available channel.

Fig. 10 shows the effects of the Zipf parameter on the amount of delay constraint violation (Fig. 10a) and the total cost (Fig. 10b). The Zipf parameter θ indicates how skewed the data access pattern is; increasing θ makes the access probability of popular data items even higher and the access probability of unpopular data items even lower. For *poach*, *cost* and *hybrid*, they generally cache popular data items more times than unpopular data items



(a) delay constraint violation (b) total cost

Fig. 8. Effects of the transmission range.



(a) delay constraint violation (b) total cost

Fig. 10. Effects of the Zipf parameter θ.

in order to reduce the aggregated access cost. Thus, increasing θ further reduces the access cost and hence reduces the total cost. From both figures, *hybrid* performs the best among the approaches satisfying the delay constraint (i.e., *delay* and *hybrid*).

10 Conclusions

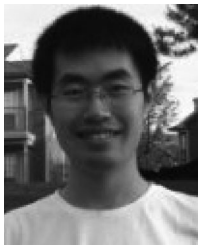
This paper studied the cache placement problem in multi-hop cognitive radio networks, which aims to minimize the total cost subject to some delay constraint. To solve the problem, we proposed three approaches: cost-based, delay-based, and hybrid. Extensive simulations demonstrated that our approaches outperform existing caching approaches in terms of total cost and delay constraint, and the hybrid approach performs the best among the approaches satisfying the delay constraint.

Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) under grant CNS-1320278.

References

[1] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury, "CRAHNs: Cognitive radio ad hoc networks," *Ad Hoc Netw.*, vol. 7, no. 5, pp. 810–836, 2009.
[2] J. Zhao and G. Cao, "Robust topology control in multi-hop cognitive radio networks," in *Proc. IEEE INFOCOM*, 2012, pp. 2032–2040.
[3] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 1, pp. 77–89, Jan. 2006.
[4] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini, "Energy-efficient caching strategies in ad hoc wireless networks," in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2003, pp. 25–34.
[5] C. C. M. Fiore, F. Mininni, and C. Chiasserini, "To cache or not to cache?" in *Proc. IEEE INFOCOM*, 2009, pp. 235–243.
[6] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative caching in wireless P2P networks: Design, implementation, and evaluation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 229–241, Feb. 2010.
[7] X. Fan, J. Cao, and W. Wu, "Design and performance evaluation of overhearing-aided data caching in wireless ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 450–463, Mar. 2013.
[8] A. W. Min, K.-H. Kim, J. P. Singh, and K. G. Shin, "Opportunistic spectrum access for mobile cognitive radios," in *Proc. IEEE INFOCOM*, 2011, pp. 2993–3001.
[9] S. Bayhan and F. Alagoz, "A Markovian approach for best-fit channel selection in cognitive radio networks," *Ad Hoc Netw.*, vol. 12, pp. 165–177, 2014.
[10] L. Musavian, S. Aissa, and S. Lambotharan, "Effective capacity for interference and delay constrained cognitive radio relay channels," *IEEE Trans. Wireless Commun.*, vol. 9, no. 5, pp. 1698–1707, May 2010.
[11] V. Asghari and S. Aissa, "Statistical QoS provisionings for wireless unicast/multicast of multi-layer video streams," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 3, pp. 420–443, Apr. 2010.
[12] C. Swamy and A. Kumar, "Primal-dual algorithms for connected facility location problems," in *Proc. Int. Workshop Approximation Algorithms Combinatorial Optimization*, 2002, pp. 256–270.
[13] 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, Mar. 2008.
[14] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Cooperative caching for efficient data access in disruption tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 611–625, Mar. 2014.
[15] J. Zhao and G. Cao, "Spectrum-aware data replication in intermittently connected cognitive radio networks," in *Proc. IEEE INFOCOM*, 2014, pp. 2238–2246.
[16] P. Wang and I. F. Akyildiz, "Can dynamic spectrum access induce heavy tailed delay?" in *Proc. IEEE Int. Symp. New Frontiers Dyn. Spectr. Access Netw.*, 2011, pp. 197–207.
[17] W. Li, X. Cheng, T. Jing, Y. Cui, K. Xing, and W. Wang, "Spectrum assignment and sharing for delay minimization in multi-hop multi-flow CRNs," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 11, pp. 2483–2493, 2013.
[18] Y. Han, E. Ekici, H. Kremo, and O. Altintas, "Throughput-efficient channel allocation in multi-channel cognitive vehicular networks," in *Proc. IEEE INFOCOM*, 2014, pp. 2724–2732.
[19] A. Abbagnale and F. Cuomo, "Leveraging the algebraic connectivity of a cognitive network for routing design," *IEEE Trans. Mobile Comput.*, vol. 11, no. 7, pp. 1163–1178, Jul. 2012.
[20] X. Huang, D. Lu, P. Li, and Y. Fang, "Coolest path: Spectrum mobility aware routing metrics in cognitive ad hoc networks," in *Proc. IEEE 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 182–191.
[21] S. Donatelli, "Kronecker algebra and (stochastic) Petri nets: Is it worth the effort," *Appl. Theory Petri Nets*, vol. 2075, no. 2001, pp. 37–56, 2001.
[22] S. M. Ross, *Introduction to Probability Models*. New York, NY, USA: Academic, 1997.
[23] G. M. L. Kou, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 1981.
[24] O. Klopfenstein, "Tractable algorithms for chance-constrained combinatorial problems," *RAIRO-Operations Res.*, vol. 43, no. 2, pp. 157–187, 2009.
[25] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation," *J. ACM*, vol. 48, no. 2, pp. 274–296, 2001.
[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[27] V. G. Kulkarni, "Shortest paths in networks with exponentially distributed arc lengths," *Networks*, vol. 16, pp. 255–274, 1986.
[28] J. Zhao, W. Gao, Y. Wang, and G. Cao, "Delay-constrained caching in cognitive radio networks," in *Proc. IEEE INFOCOM*, 2014, pp. 2094–2102.
[29] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. IEEE Workshop Mobile Comput. Syst. Appl.*, 1999, pp. 90–100.
[30] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, 1999, pp. 126–134.



Jing Zhao received the BE degree in computer science and technology from Tsinghua University in 2009. He is currently working toward the PhD degree in computer science and engineering at the Pennsylvania State University. His research interests include cognitive radio networks, network reliability, and robustness. He is a student member of the IEEE.

Wei Gao received the BE degree in electrical engineering from the University of Science and Technology of China in 2005 and the PhD degree in computer science from Pennsylvania State University in 2012. He is currently an assistant professor in the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville. His research interests include wireless and mobile network systems, mobile social networks, cyberphysical systems, and pervasive and mobile computing. He is a member of the IEEE.

Yi Wang received the BS degree in math from Tsinghua University in 2005, the MS degree in computer science from the Chinese Academy of Sciences in 2008, and the PhD degree in computer science and engineering from the Pennsylvania State University in 2013. His research interests include wireless networks and mobile computing, camera sensor networks, and wireless ad hoc networks.

Guohong Cao received the BS degree in computer science from Xian Jiaotong University and the PhD degree in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering, Pennsylvania State University, where he is currently a professor. He has published more than 200 papers in the areas of wireless networks, wireless security, vehicular networks, wireless sensor networks, cache management, and distributed fault tolerant computing. He has served on the editorial boards of the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Wireless Communications*, *and IEEE Transactions on Vehicular Technology*, and has served on the organizing and technical program committees of many conferences, including the TPC chair/co-chair of IEEE SRDS2009, MASS2010, and INFOCOM2013. He received the US National Science Foundation (NSF) CAREER award in 2001. He is a fellow of the IEEE.