Delay-Constrained Caching in Cognitive Radio Networks

Jing Zhao*, Wei Gao†, Yi Wang* and Guohong Cao*
*Pennsylvania State University, University Park, PA
†University of Tennessee, Knoxville, TN
* {juz139,yuw124,gcao}@cse.psu.edu, † weigao@utk.edu

***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. To the best of our knowledge, we are the first to use 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 three approaches: cost-based, delay-based, and hybrid. 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.**

I. 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], [2], 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 [3]. 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 A wants to access the data generated at node B which is faraway. Without caching, the data has to travel multiple hops to reach A. 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 B's data is cached/replicated at multiple nodes in the network, A 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 [4], [5], [6] model the access delay by the number of hops required to obtain the data, and assume all links have equal transmission delay. Some others [7], [8] assume the link transmission delay can be calculated accurately. However, in cognitive radio networks, the link transmission delay depends on the primary user appearance, and it will be much longer when primary users appear.

To overcome the aforementioned difficulty, we model the primary user appearance as a continuous-time Markov chain following several prior works [9], [10], 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* [11], [12], i.e., the data access delay is statistically bounded.

Although the delay constraint can be satisfied by caching data at all nodes, it is inefficient due to the cost which is affected by different factors. For example, disseminating data to the 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. Thus, we should design better caching algorithms to reduce various types of cost while satisfying the delay constraint. The contributions of the paper are three folds.

- We are the first to use 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

This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-1320278 and by Network Science CTA under grant W911NF-09-2-0053.

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 II reviews related work. In Section III, we provide an overview of our work. We define the system models in Section IV, and then formulate our problem in Section V. Sections VI, VII and VIII present the three caching approaches in detail. We show simulation results in Section IX, and conclude the paper in Section X.

II. 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* [13], which is known to be NP-hard. The problem can be solved by using a greedy algorithm *poach* [5], 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 [4], [6], [7], [8], [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 et al [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 routing protocol [17] has been proposed to select paths with the smallest transmission delay, but the selected routing path may frequently change due to primary user appearance. To address this problem, some protocols [18], [19] 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.

III. 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. Each data item is generated by the data source and cached at appropriate nodes, so that queries from the requesters can be answered by the caching nodes directly with less delay. The data is updated periodically, and thus the data source needs to disseminate the updated data to the caching nodes. The focus of this paper is hence on determining appropriate caching nodes.

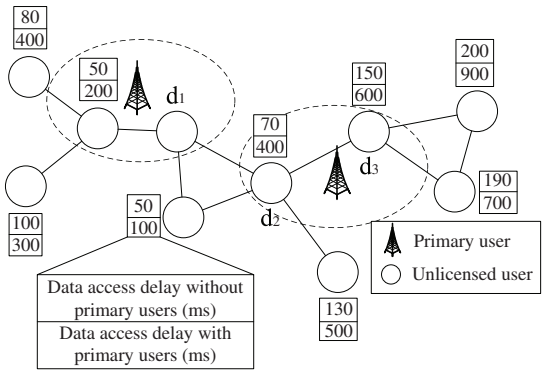


Fig. 1. Delay-constrained caching for data item d_1. The data access delay is at 90% confidence interval, i.e., the actual data access delay has 90% probability to be less than this value.

Figure 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 presence of primary users which affect 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 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 V.

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.

IV. 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 [9], [10], the primary user appearance is modeled as the following continuous-time Markov chain, based on which

the link transmission delay is probabilistically estimated.

A. Primary User Appearance

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 $\bigoplus_{c=1}^{\mathcal{C}} Q_{e,c}$ is the generator matrix for Markov chain $\mathbf{M}_e(t)$ [20]. Here $\bigoplus$ 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. $A \bigoplus B = A \bigotimes I_p + I_n \bigotimes B$.

The operation $\bigotimes$ 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. $C = A \bigotimes B$ denotes that $C_{ij} = A_{i_1 j_1} B_{i_2 j_2}$ where $i = (i_1 - 1)p + i_2$ and $j = (j_1 - 1)q + j_2$.

B. Link Transmission Delay

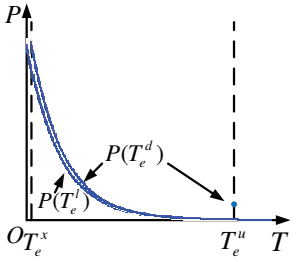


Fig. 2. Approximation of link transmission delay

The basic idea for modeling the link transmission delay is as follows. Suppose node v wants to send data over link $e = (u, v)$. If all licensed channels are currently accessed by primary users, node v may wait until some licensed channel becomes available, or switch to an unlicensed channel for data transmission. Generally speaking, if the appearance of primary users lasts a long time (e.g., TV transmitters), the link should switch to an unlicensed channel; otherwise, the link should wait for an available licensed channel. Therefore, the transmission delay of link e, denoted by T_e^d, can be modeled as follows.

$$T_e^d = T_e^w + T_e^x = \begin{cases} T_e^l + T_e^x & T_e^l < T_e^u \\ T_e^u + T_e^x & otherwise \end{cases}$$

where T_e^w is the time that link e must wait (switch) for an available channel, T_e^x is the time for actual data transmission, T_e^l is the time that link e waits until some licensed channel becomes available, and T_e^u is the time that link e switches to an unlicensed channel. Note that the link transmission delay is bounded ($T_e^d \leq T_e^u + T_e^x$), so there still exists persistent network connection. This differs from Disruption Tolerant Networks (DTNs) [21] in which nodes are only intermittently connected, and hence it is unsuitable to use delay tolerant networking approaches in our scenario.

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

In Figure 2, $P(T_e^d)$ shows the distribution of link transmission delay. In order to formally derive the distribution of data access delay in Section VII, we approximate $P(T_e^d)$ by exponential distribution $P(T_e^l)$ with the following assumptions. We assume large data items can be split into smaller ones to make T_e^x very small. Another observation is that, due to the nature of exponential distribution, $P(T_e^l)$ approaches zero very quickly as T_e^l increases, so there is generally little chance for T_e^l to be much larger than T_e^u. Therefore, $P(T_e^d)$ is very close to $P(T_e^l)$ so $P(T_e^l)$ can be used to approximate $P(T_e^d)$.

V. 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.

A. Cost

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.

2) Access Cost: We define the access cost as the aggregated data access delay, following several prior caching works [4], [6]. Let d_{uv} be the expected transmission delay between nodes u and v, which is 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 $\sum_v p_v D_v$, where p_v is the data access probability of node v (the probability that node v accesses the data), and $D_v = 2\min_{u\in\mathcal{V}} d_{vu}$ indicating the expected round-trip transmission delay from v to the nearest caching node.

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.

B. Delay Constraint

Let α be the confidence level, and β be the delay threshold. For node v, the delay constraint is defined as $P(a_v \le \beta) \ge \alpha$, where a_v is the data access delay of node v and is approximated by D_v (the expected round-trip transmission delay from v to the nearest caching node) in the access cost.

C. Delay-constrained Caching Problem

The cache placement problem in cognitive radio networks can be formulated as the following delay-constrained caching problem.

$$\textbf{minimize } \sum_v p_v D_v + \mathcal{W} b L(\mathcal{G})$$
$$\textbf{subject to } P(a_v \le \beta) \ge \alpha, \forall v$$

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* [5] 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.

VI. Cost-Based Approach

In this approach, we formulate the problem as the following linear programming problem to minimize the total cost. Here v_1 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 V). $z_e = 1$ if e is in $\mathcal{G}$; otherwise, $z_e = 0$. $\delta(S)$ is the set of edges who have only one end-node belonging to S. d_{uv} is the expected transmission delay between nodes u and v (defined in Section V). 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 an edge in the dissemination graph.

$$\textbf{minimize } \sum_{u\in V} p_u \sum_{v\in V} d_{uv} x_{uv} + \mathcal{W} b \sum_{e\in E} z_e \quad (1)$$
$$\textbf{subject to}$$
$$\sum_{v\in V} x_{uv} = 1, \forall u \in V \quad (2)$$
$$\sum_{v\in S} x_{uv} \le \sum_{e\in\delta(S)} z_e, \forall S \subseteq V \setminus \{v_1\}, \forall u \in V \quad (3)$$
$$x_{uv}, z_e \in \{0, 1\} \quad (4)$$

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 demand 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 address our problem using an existing approach [13] for the connected facility location problem.

Note that the delay constraint cannot be contained in the aforementioned linear programming problem, since a linear programming problem 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.

VII. 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.

A. 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 length equal to the corresponding link transmission delay, which is an exponentially distributed random variable. Let l_{uv} be the length of the shortest u-v path. If the query is allowed to be 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 $\min_{v\in\mathcal{V}} l_{uv}$. Here $\min_{v\in\mathcal{V}} l_{uv}$ is the minimum of all l_{uv}'s in which v is a caching node (the data source is regarded as

a caching node). If we obtain the distribution of $\min_{v\in\mathcal{V}} l_{uv}$, the distribution of u's data access delay is obtained.

In the literature, Kulkarni has proposed an algorithm [25] to derive the distribution of the shortest path length from one node to another node, in a graph where the edge lengths are 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.

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.

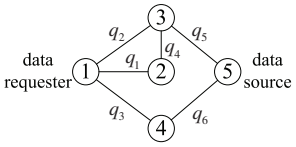


Fig. 3. An example of a data source. Each edge is associated with the transmission delay represented by an exponential distribution. The label associated with each link represents the parameter of the distribution.

Let $\mathcal{N}(t)$ be the set of *disabled nodes* at time t. The disabled nodes are the nodes that are useless for the propagation of the query towards the data source. For example, in Figure 3, suppose a query is broadcast from node 1 at time 0 when nodes 1-2 and nodes 1-4 do not share any common channel due to primary user appearance. Since nodes 2 and 4 have to wait some time before receiving the query, the query first reaches node 3 before it reaches nodes 2 and 4. After node 3 broadcasts the query, it becomes disabled. Node 2 is also disabled even if node 2 has not received the query. This is because any query sent from node 2 cannot reach any nodes that are not disabled. Since node 2 is useless for the propagation of the query towards the data source (node 5), node 2 is a disabled node.

TABLE I
STATE SPACE FOR THE EXAMPLE NETWORK

State	State Description
1	{1}
2	{1,2}
3	{1,4}
4	{1,2,4}
5	{1,2,3}
6	{1,2,3,4}
7	{1,2,3,4,5}

According to [25], $\{\mathcal{N}(t), t \geq 0\}$ is a continuous-time Markov chain. The state space for the example graph (Figure 3) is shown in Table I, and the generator matrix Q is shown in Table II. Let S_i be the ith state, and n be the number of states. In this example, $n = 7$. If the state is the final state S_n, the data source (node 5) receives the message, and ends the aforementioned stochastic process. Let T be the the shortest

TABLE II
GENERATOR MATRIX FOR THE EXAMPLE NETWORK

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

path length from the requester to the data source. We have

$$T = \min\{t \geq 0 : \mathcal{N}(t) = S_n | \mathcal{N}(0) = S_1\}$$

Let $F(t)$ be the cumulative distribution function of T, i.e., $F(t) = P(T \leq t)$. $F(t)$ 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 $k \to \infty$.

2) Distribution of the Shortest Path Length from a Data Requester to Multiple Caching Nodes: We use an example graph (Figure 4) to show how to derive the distribution of the shortest path length from a data requester to multiple caching nodes. If node 4 also caches the data, the aforementioned stochastic process should be adjusted as follows. That is, if either node 4 or node 5 receives the message, the process can be ended and thus states 3, 4, 6, 7 of Table I are merged into one final state. The updated state space is shown in Table III, and the corresponding generator matrix is shown in Table IV. Then the distribution function of the shortest path length can be calculated accordingly.

B. 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. The data access delay is twice of the shortest path

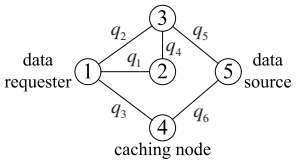


Fig. 4. An example of multiple caching nodes, where each edge is associated with the transmission delay represented by an exponential distribution.

TABLE III
STATE SPACE FOR THE EXAMPLE NETWORK (FIGURE 4) WITH NODE 4 AS A CACHING NODE

State	State Description
1	{1}
2	{1,2}
3	{1,2,3}
4	{1,2,3,4,5}

length from the requester to the caching nodes (i.e., $2T$). The delay constraint can be denoted by $P(2T \leq \beta) \geq \alpha$, or equivalently,

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

Since it is impossible to get the exact value of $F(t)$, we use the approximation $F_k(t)$ to check the delay constraint. 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.

C. Algorithm Description

We check whether the delay constraint is satisfied for each node. If some nodes do not satisfy the delay constraint, the node violating the delay constraint the most is selected for caching the data. Here we use $\alpha - F_v^{\mathcal{V}}(\beta/2)$ ($F_v^{\mathcal{V}}(t)$ denotes $F(t)$ (see Section VII-B), with v as a requester and $\mathcal{V}$ as the set of caching nodes) 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 VII-B) as an approximation. We check again whether the delay constraint is satisfied for all other nodes. If not, the aforementioned procedure is repeated. Otherwise, we have selected enough caching nodes.

VIII. 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.

A. Data Structure

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.

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 [26], 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.

TABLE IV
GENERATOR MATRIX FOR THE EXAMPLE NETWORK (FIGURE 4) WITH NODE 4 AS A CACHING NODE

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

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.

B. 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 1 constructs graph G_1 as shown in Figure 5. Here nodes 4 and 5 are in node 1's caching node list. The caching path for node 4 is 1-4, and the caching path for node 5 is 1-3-5.

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 VII-A2 to G_v.

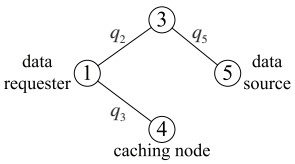


Fig. 5. The graph for calculating the data access delay of node 1. Nodes 4 and 5 are in the caching node list.

C. 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.

1) Caching Node Selection: Each node checks whether the delay constraint is satisfied. If the constraint is satisfied, nothing is done. Otherwise, the node requests to be a caching node.

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, if all its neighbors with higher average access delay satisfy the delay constraint, it may request to be a caching node; otherwise, the node must wait until all the neighbors with higher average access delay satisfy the delay constraint.

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., $\Delta_v = p_v D_v - \mathcal{W} b h_v$. Here p_v is node v's access probability, D_v is v's average data access delay, b is the amount of traffic rate for data dissemination (defined in Section V), and h_v is the number of hops from node v to the nearest caching node in the caching node list. If $\Delta_v > 0$, v requests to be a caching node. Otherwise, nothing is done.

2) Update: Suppose node v is informed that u becomes a new caching node. 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 (selected following Section VIII-A). 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) $|N_v| < \gamma$: we calculate the v-u path with minimum expected transmission delay. Node u is added to N_v^u, and the calculated path is assigned to P_v^u.

(ii) $|N_v| = \gamma$: node v decides whether u should replace some caching node in N_v. Let u' be the caching node with the longest expected transmission delay from v. Node v checks whether replacing u' by u leads to any improvement.

Specifically, v calculates the v-u path with the minimum expected transmission delay (selected following Section VIII-A1), and checks whether the expected transmission delay of the new v-u path is less than that of the former v-u' path. If so, the replacement is performed. Otherwise, v does not perform the replacement.

IX. Performance Evaluations

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

A. Simulation Setup

In our simulations, we randomly place 25 nodes in a 3200×300 m^2 area. The transmission range is 250 m. If two nodes are within the transmission range, there is a link between them.

We use the same data query model as in previous studies [4], [6], [7]. Each node generates a stream of queries. The query generation time follows exponential distribution with mean value 5s. 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 [27].

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, ... 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 $(id + n \mod i) \mod n$, where n is the database size ($n = 100$ in our simulation).

The data items are generated by two nodes, node 0 and node 1; data items with even id's are generated by node 0 and the rests are generated by node 1. The data size is uniformly distributed between 100B and 7KB. The data update interval follows exponential distribution with mean value 10s.

In our system model, the link transmission delay depends on the primary user appearance, and is approximated by exponential distribution. The primary user appearance is location-dependent; nodes around the same location are affected by similar primary users. For each grid i, the distribution parameter q_i is assigned a random variable between 0.001 and 0.01, corresponding to the mean waiting period between 100ms and 1s. For each link $e = (u, v)$, if u is in grid i and v is in grid j, the distribution parameter is $\min\{q_i, q_j\}$.

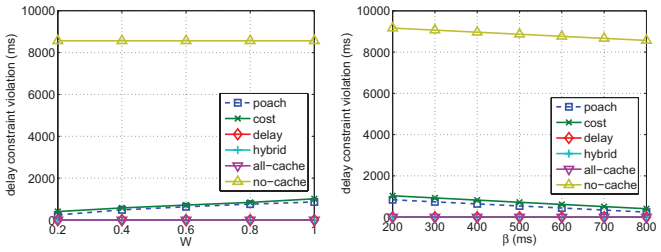
To check the delay constraint, we use confidence level $\alpha = 0.9$. The delay threshold β is different for different experiments. The cost ratio $\mathcal{W}$ is also adjusted to study its effect on performance.

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* [5] 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* in which data is cached at all nodes, and *no-cache* in which no node caches the data.

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% 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.

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



(a) effect of $\mathcal{W}$ ($\beta = 800$ms) (b) effect of β ($\mathcal{W} = 0.2$ms/bps)

Fig. 6. The amount of delay constraint violation

Figure 6(a) shows the effect of $\mathcal{W}$ on the amount of delay constraint violation. *no-cache* has the highest amount of delay constraint violation 8565ms 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 239ms (404ms) to 859ms (1011ms). 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.

Figure 7 shows the effect of $\mathcal{W}$ on the total cost, and Figures 8, 9 show the effect of $\mathcal{W}$ on the access cost and the dissemination cost, respectively.

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% 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% higher total cost than *cost*. *delay* does not minimize the total cost, and it has 5%-29% higher total cost than *hybrid*. *all-cache* and *no-cache* have the highest total cost.

Access cost and **Dissemination cost:** For the delay-based approach (*delay*), the access cost (the dissemination cost) stays flat as $\mathcal{W}$ increases, since it only considers the delay constraint and is hence unaffected by $\mathcal{W}$. The access cost is the highest among all non-naive approaches since the number of caching nodes to be placed is only enough to satisfy the delay constraint. Note that the access cost can be further reduced by placing more caching nodes along the path used for data dissemination, without increasing the dissemination cost. This explains why the other three non-naive approaches (*poach, cost, hybrid*) have similar dissemination cost but much lower access cost compared to the delay-based approach (*delay*).

For the other three non-naive approaches, the access cost increases as $\mathcal{W}$ increases. When the dissemination cost is given more weight, the number of caching nodes (the dissemination cost) should be reduced so that the total cost can increase less rapidly. As a result, the dissemination cost decreases while the access cost increases.

The approach *all-cache* (*no-cache*) has the highest dissemination cost (the highest access cost) among all approaches. Note that the access cost of *all-cache* (the dissemination cost of *no-cache*) is zero although not shown in the figures.

C. Effects of the Delay Threshold (β)

Figure 6(b) 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 200ms to 800ms, the amount of delay constraint violation decreases from 839ms (1004ms, 9165ms) to 239ms (404ms, 8565ms). *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.

Figure 10 shows the effect of β on the total cost. Figures 11 and 12 show the effect of β on the access cost and the dissemination cost, respectively.

For *poach*, *cost*, *all-cache*, and *no-cache*, the total cost (the access cost, the dissemination 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 minimizing the total cost is the main goal. The total cost of *cost* is 35% (84%) lower than that of *all-cache* (*no-cache*), and 6% 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 200ms to 800ms. Increasing β relaxes the delay constraint, so less nodes are needed to cache the data. Since *delay* does not minimize the total cost, the access cost increases by 372% and it outweighs 6% decrease of dissemination cost, leading to 19% increase of the total cost.

For *hybrid*, the total cost decreases by 5% as β increases from 200ms to 800ms. 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% lower total cost than *delay*.

X. Conclusions

This paper studied the cache placement problem in multi-hop cognitive radio networks, which aims to minimize the

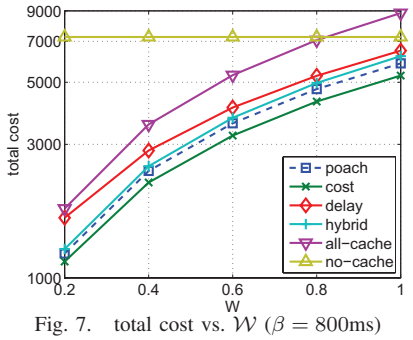


Fig. 7. total cost vs. $\mathcal{W}$ ($\beta = 800$ms)

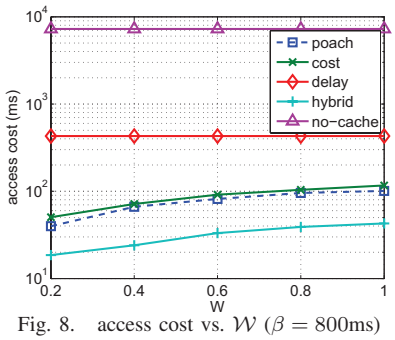


Fig. 8. access cost vs. $\mathcal{W}$ ($\beta = 800$ms)

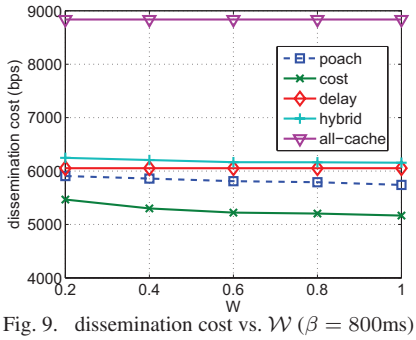


Fig. 9. dissemination cost vs. $\mathcal{W}$ ($\beta = 800$ms)

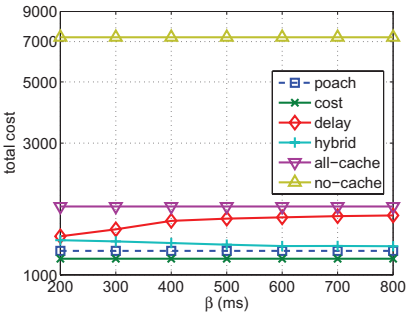


Fig. 10. total cost vs. β ($\mathcal{W} = 0.2$)

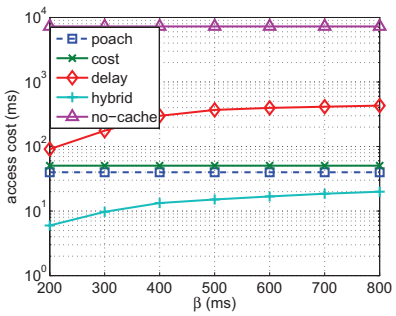


Fig. 11. access cost vs. β ($\mathcal{W} = 0.2$)

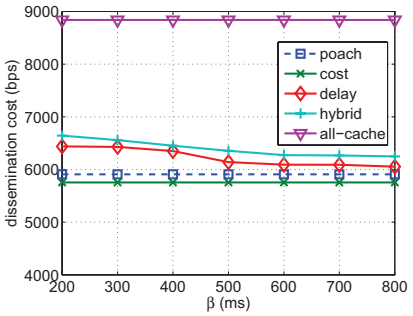


Fig. 12. dissemination cost vs. β ($\mathcal{W} = 0.2$)

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.

REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury, "CRAHNs: Cognitive radio ad hoc networks," *Ad Hoc Networks*, vol. 7, no. 5, pp. 810–836, 2009.

[2] A. M. Wyglinski, M. Nekovee, and T. Hou, *Cognitive Radio Communications and Networks: Principles and Practice*. Academic Press, 2009.

[3] J. Zhao and G. Cao, "Robust Topology Control in Multi-hop Cognitive Radio Networks," in *IEEE INFOCOM, 2012*.

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

[5] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini, "Energy-Efficient Caching Strategies in Ad Hoc Wireless Networks," in *ACM MobiHoc, 2003*.

[6] B. Tang, H. Gupta, and S. R. Das, "Benefit-Based Data Caching in Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 3, pp. 289–304, 2008.

[7] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative Caching in Wireless P2P Networks: Design, Implementation, and Evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 229–241, 2010.

[8] X. Fan, J. Cao, and W. Wu, "Design and Performance Evaluation of Overhearing-aided Data Caching in Wireless Ad Hoc Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 450–463, 2013.

[9] A. W. Min, K.-H. Kim, J. P. Singh, and K. G. Shin, "Opportunistic Spectrum Access for Mobile Cognitive Radios," in *IEEE INFOCOM, 2011*.

[10] S. Bayhan and F. Alagoz, "A Markovian approach for best-fit channel selection in cognitive radio networks," *Ad Hoc Networks*, 2011.

[11] L. Musavian, S. Aissa, and S. Lambotharan, "Effective capacity for interference and delay constrained cognitive radio relay channels," *IEEE Transactions on Wireless Communications*, vol. 9, no. 5, pp. 1698–1707, 2010.

[12] V. Asghari and S. Aissa, "Statistical QoS Provisionings for Wireless Unicast/Multicast of Multi-Layer Video Streams," *IEEE Journals on Selected Areas in Communications*, vol. 28, no. 3, pp. 420–443, 2010.

[13] C. Swamy and A. Kumar, "Primal-Dual Algorithms for Connected Facility Location Problems," in *International WOrkshop on Approximation Algorithms for Combinatorial Optimization (APPROX), 2002*.

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

[15] J. Zhao and G. Cao, "Spectrum-Aware Data Replication in Intermittently Connected Cognitive Radio Networks," in *IEEE INFOCOM, 2014*.

[16] P. Wang and I. F. Akyildiz, "Can Dynamic Spectrum Access Induce Heavy Tailed Delay?" in *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), 2011*.

[17] G. Cheng, W. Liu, Y. Li, and W. Cheng, "Spectrum Aware On-demand Routing in Cognitive Radio Networks," in *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), 2007*.

[18] A. Abbagnale and F. Cuomo, "Leveraging the Algebraic Connectivity of a Cognitive Network for Routing Design," *IEEE Transcations on Mobile Computing*, vol. 99, 2011.

[19] X. Huang, D. Lu, P. Li, and Y. Fang, "Coolest Path: Spectrum Mobility Aware Routing Metrics in Cognitive Ad Hoc Networks," in *IEEE ICDCS, 2011*.

[20] S. Donatelli, "Kronecker algebra and (stochastic) Petri nets: Is it worth the effort," *Application and Theory of Petri Nets*, vol. 2075, no. 2001, pp. 37–56, 2001.

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

[22] S. M. Ross, *Introduction to Probability Models*. Academic Press, 1997.

[23] A. Agrawal, P. Klein, and R. Ravi, "When trees collide: An approximation algorithm for the generalized Steiner problem on networks," *SIAM Journal on Computing*, vol. 24, no. 3, pp. 440–456, 1995.

[24] O. Klopfenstein, "Tractable algorithms for chance-constrained combinatorial problems," *RAIRO - Operations Research*, vol. 43, no. 2, pp. 157–187, 2009.

[25] V. G. Kulkarni, "Shortest Paths in Networks with Exponentially Distributed Arc Lengths," *Networks*, vol. 16, pp. 255–274, 1986.

[26] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *IEEE Workshop on Mobile Computing Systems and Applications, 1999*.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *IEEE INFOCOM, 1999*.