Probabilistic Cooperative Caching-Based Content Subscribing in Opportunistic Mobile Networks

T^{2}WISTOR: TTU Wireless Mobile Networking Laboratory
Technical Report TR-01-2017

Bitao Peng[1], Sunho Lim[2], Byungkwan Jung[2], Cong Pu[3], Xinying Qiu[1], and Manki Min[4]

[1] Cisco School of Informatics, Guangdong University of Foreign Studies, China
pengbitao@gdufs.edu.cn, 201010021@oamail.gdufs.edu.cn
[2] Dept. of Computer Science, Texas Tech University, Lubbock, TX 79409
{sunho.lim, byung.jung}@ttu.edu
[3] Weisberg Division of Computer Science, Marshall University, Huntington, WV 25755
puc@marshall.edu
[4] Dept. of Electrical Engineering and Computer Science, South Dakota State University, Brookings, SD 57007
manki.min@sdstate.edu

Abstract. Publish/subscribe (pub/sub) based data access techniques have been actively researched in opportunistic mobile networks (OppNets) that can be characterized by the intermittent connectivity and limited network capacity. Because of time-varying network topologies due to the node mobility, it becomes an issue to guarantee end-to-end path for seamless communication. In this paper, we propose a probabilistic cooperative caching based content subscribing scheme, called *PCC*, to enhance data availability and accessibility in OppNets. We first present the basic pub/sub based operations and their corresponding information to be maintained and exchanged. Then we measure the expected data delivery latency as a basis for the data utility and define a data utility function based on the time-to-live of data and data delivery latency. We further refine exchanging the data using a multi-objective linear programming and propose a heuristic approach to cooperatively cache the data among nodes. We conduct extensive simulation experiments for performance comparison using the opportunistic network environment simulator (ONE). The simulation results show that our scheme outperforms other three existing schemes (i.e., Ad hoc Podcasting, Content Place, and Least TTL) in terms of delay, number of delivered data, and number of hops, and indicate that our scheme can be a viable approach in OppNets.

1 Introduction

Opportunistic mobile networks (OppNets) characterized by intermittent connectivity and limited network capacity can be categorized into delay tolerant networks (DTNs) [1]. As mobile and wireless devices (later in short, nodes) equipped with the advanced communication capability become increasingly popular and prevalent, OppNets are rapidly emerging as an alternative to conventional infrastructure-based communication. To enhance data availability and accessibility, nodes often deploy a *publish/subscribe* (pub/sub) based scheme to exchange data upon encountering one another. In the pub/sub, nodes play a role as a publisher, a subscriber, or an intermediate forwarder. A publisher advertises its data subject and produces data item but a subscriber requests and consumes data based on its own interests. When nodes are encountered one another, the publisher can deliver data to the subscriber directly or indirectly through multi-hop relays using intermediate forwarders. Due to the mobility of nodes, however, it is not trivial to guarantee end-to-end path for communication in constantly varying network topologies.

In light of this, we propose a probabilistic cooperative caching-based content subscribing scheme, called *PCC*. The PCC is a pub/sub based data dissemination approach that considers the inter-contact time, data utility, and data freshness. The basic idea is that whenever nodes encounter one another, they exchange the data that can maximize the data utility based on the data freshness and data delivery latency. Under the constraint of storage space, nodes further optimize to cache the data. Our contribution is summarized in three-fold:

- First, we present three basic pub/sub based operations and their corresponding information to be maintained and exchanged for encountered nodes, in terms of subscribing channels, publishing data, and delivering subscribed data.
- Second, we develop three major operations to realize the PCC: expected data delivery latency, data utility function, and data utility optimization. We measure the expected data delivery latency as a basis to compute the data utility. We also define a data utility function to judiciously exchange the data and maximize the data utility based on the data freshness and data delivery latency. We further optimize exchanging the data using a multi-objective linear programming and propose a heuristic approach for cooperative caching to efficiently access the data directly and indirectly.
- Third, we revisit three existing schemes (i.e., *Ad hoc Podcasting* [2], *ContentPlace* [3], and *Least TTL*), modify them to work in our testbed, and evaluate them with the PCC for performance comparison.

We conduct extensive simulation experiments using the opportunistic network environment simulator (ONE) [4] for performance evaluation and multi-dimensional analysis. Simulation results show that the PCC outperforms other existing schemes in terms of average packet delay, number of delivered packets, and average number of hops.

The rest of paper is organized as follows. Section 2 reviews the related work. Section 3 presents the system model and proposed approach. Section 4 devotes comprehensive performance evaluation and analysis. Finally, Section 5 concludes the paper with future work.

2 Related Work

Opportunistic data forwarding is a variant of epidemic routing, where nodes exchange data which are not in common when they encounter one another in intermittently connected networks [5]. In a spray-and-wait based approach [6], nodes only forward a self-limited number of data to reduce the delay but improve the energy efficiency. Since nodes often move and follow a mobility pattern, [7] predicts a future node contact based on the frequency and history of contacts to select the best data forwarder. [8] calculates the direct and indirect subscribe values of data based on the contact history to maximize the content inventory. With users' social environment structure, it computes the relevance of data for users and their social communities and then proposes a social-aware data forwarding strategy. [9] proposes a pub/sub based forwarding scheme by considering social behavior in OppNets, where a single broker is selected in the community to center information. Unlike prior a single-layer social network, [10] defines a multi-layer social network model by combining both on-line and off-line social relationships to decide the data forwarding in OppNets.

In opportunistic data access, a pub/sub based scheme and its variants are often deployed, where each subscriber and publisher declares its own interests and registers its data, respectively. The PodNet project [11, 12] and Content Place [3] support users to subscribe a list of interests as well as to exchange the data based on a pair-wise opportunistic contact. [8] deploys a Tit-For-Tit scheme and proposes an incentive-driven strategy to deal with a selfish pub/sub in OppNets. [13] investigates a broker node that forwards the data to nodes according to their subscriptions.

Data utility has been used as a key index in opportunistic data management. Each node caches data in a local storage based on the estimated utility [8, 14]. [8] combines both direct and indirect subscribed data and considers the probability of other nodes having a data copy to compute the data utility. [14] also considers a ranked search to decide which data will be exchanged based on how strongly other nodes desire for the data. Cooperative caching is an alternative way to improve data availability and accessibility as well as to reduce the access delay by storing a subset of data over multiple nodes. [15] presents a hierarchical cooperative caching by virtually dividing a storage into three parts (i.e., self, friends, and strangers) and considers the relationship between pair-wise nodes to differentiate caching strategies.

In summary, diverse approaches have been proposed but little attention has been paid for a probabilistic cooperative caching-based content subscribing in the realm of OppNets.

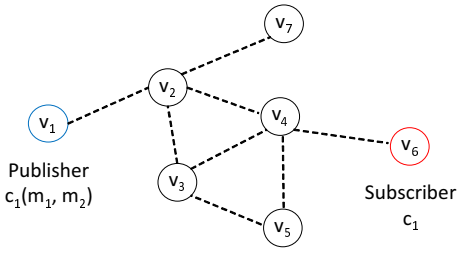


Fig. 1. A snapshot of simple pub/sub based communication in an OppNet, where v_1 publishes the data (e.g., m_1 and m_2) that belong to a channel c_1 what v_6 subscribes. When v_1 opportunistically encounters v_2, it can send both m_1 and m_2 to v_6 through multi-hop relays, e.g., forwarded by v_2 and v_4.

3 Probabilistic Cooperative Caching based Content Subscribing

In this section, we first present a system model and then propose a probabilistic cooperative caching-based content subscribing scheme, called PCC.

3.1 Pub/Sub based System Model

We deploy a pub/sub based communication model, where a set of nodes is uniformly distributed in an OppNet. Each node can independently produce data as a publisher and request and consume data as a subscriber as well. In this paper, we assume that published data can be classified into one of the subjects what node is interest in, called *channel*, such as real-time traffic information, stock, weather, music, etc. Nodes may request a set of channels that they are interested in but may not know what channels other nodes publish. We also assume that once a node requests the set of channels then it will not change them during the communication. For example, Fig. 1 shows a snapshot of simple pub/sub operation in an OppNet that can be represented as an undirected graph $G(V, E)$, where $V = \{v_i\}$, $E = \{e_{i,j} \mid (v_i, v_j) \wedge i \neq j\}$, and i and j range from 1 to n, respectively. Here, n is the total number nodes in the network. Without loss of generality, we set a weight $w_{i,j}$ to every edge $e_{i,j}$ as an average inter-contact time between two nodes based on the contact history. In this model, we present three basic pub/sub based operations: (i) subscribing channels; (ii) publishing data; and (iii) delivering subscribed data. In subscribing channel, when nodes encounter one another, each node exchanges its own channel lists containing what it is interested in and what it has collected from prior contacted nodes. Each node maintains a subscription table (ST), where each entry consists of two components: node *id* and a set of subscribed channels, $ST = [v_{id}, \{c_{id}\}]$. For example, v_i initially has a single entry $[v_i, \{c_i\}]$ in the ST, where $\{c_i\}$ is a set of subscribed channels what v_i is interested, and the table will gradually have more entries as v_i encounters other nodes. When v_i encounters v_j, each node exchanges its channel lists. If v_i already has the channel list of v_j (e.g.,

Notations:

• $ST_a[v_{id}, \{c_{id}\}]$: A subscription table in a node v_a, where v_{id} and $\{c_{id}\}$ are a channel subscribed node id and a set of subscribed channel ids, respectively.

• $DT_a[m_{id}, c_{id}, t]$: A data digest table in a node v_a, where m_{id}, c_{id}, and t are a data id, a channel id, and a time-to-live of m_{id}, respectively.

• $DDT_a[m_{id}, c_{id}, g_{id}, t]$: A delivered data digest table in a node v_a, where m_{id}, c_{id}, g_{id}, and t are a data id,node id, channel id, and a time-to-live of m_{id}, respectively.

• $SUT_a[m_{id}, ut, size]$: A sorted utility table in a node v_a, where ut and $size$ are the data utility and size of data, m_{id}, respectively.

• $pkt[Req, nid, m_{id}]$: A data request packet sent from a node v_{nid} for data m_{id}.

◇ When v_a encounters v_b:

if $ST_a \neq ST_b$

$ST_a = ST_a \cup (ST_b - ST_a)$;

for each $c_{id'} \in ST_a[v_b, \{c_{id}\}]$

for each $c_{id''} \in DT_b[m_{id}, c_{id}, t]$

if $c_{id'} == c_{id''}$

Forward $pkt[Req, a, m_{id}]$ to v_b;

◇ When v_b receives $pkt[Req, a, m_{id}]$ from v_a:

Forward m_{id} to v_a and update DDT_b;

◇ When v_a (or v_b) exchanges DDT_a (or DDT_b) with v_b (or v_a) :

v_a (or v_b) computes ut of SUT_a (or SUT_b) using Eq. 9;

Fig. 2. The pseudo code of the basic pub/sub based operations.

$\{c_j\}$), it does not update the table. In publishing data, when v_i publishes data (m), it sends a *Data* packet containing data *id*, published channel *id* that the published data belongs to, and time-to-live (t), e.g., $pkt[m_{id}, c_{id}, t]$. In delivering subscribed data, after exchanging the channel lists, nodes search their local cache and deliver corresponding data. Each node maintains a delivered data digest table (DDT), where each entry consists of four components: delivered data *id*, channel *id*, neighbor node *id*, and time-to-live (t), $DDT = [m_{id}, c_{id}, g_{id}, t]$. Here, the data is valid during the t period but its corresponding entry in the DDT will be removed from the cache after the t expires. Each node encountered also sends the DDT to its neighbor nodes. Thus, nodes ultimately gather the information about which data have been delivered to which nodes. In addition, each node maintains a data digest table (DT) to exchange with encountered nodes. The basic pub/sub based operations are summarized in Fig. 2.

3.2 Proposed Approach

The PCC is realized by conducting three major operations: (i) expected data delivery latency; (ii) data utility function; and (iii) data utility optimization.

Expected Data Delivery Latency We first measure the expected data delivery latency as a basis to develop a data utility function and a data utility based

caching technique. The basic idea is that the expected data delivery latency can be approximated by adding the inter-contact time of nodes located within the path between source and destination nodes.

In this paper, we consider a weight $w_{i,j}$ between v_i and v_j is an average inter-contact time, which is assumed to follow an exponential distribution and has been shown and analyzed by many prior works[16]. Thus, the contact between v_i and v_j remains a Poisson process with a contact frequency $\lambda_{i,j}$. Suppose a continuous random variable X denotes the inter-contact time. Then its probability density function can be expressed as,

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{1}$$

For example, a source node (v_1) sends a data packet to a destination node (v_n) and the data is relayed by intermediate nodes, v_k, where $k = 2$ to $n-1$. We denote X_i as the inter-contact time between nodes v_i and v_{i+1}, and then the data delivery latency from v_1 to v_n can be expressed using a continuous random variable Y below,

$$Y = X_1 + X_2 + \ldots + X_{n-1}. \tag{2}$$

Here, Y is a hypo-exponential random variable and its probability density function can be expressed as,

$$f_Y(x) = \sum_{i=1}^{n-1} C_{i,n-1} \lambda_i e^{-\lambda_i x}, \tag{3}$$

where

$$C_{i,n-1} = \prod_{i \neq j} \frac{\lambda_j}{\lambda_j - \lambda_i}.$$

We also consider both direct and indirect data transfers through multi-hop relays to measure the data delivery latency. For example, a source node (v_i) can directly send a data packet to a destination node (v_n) when it is encountered. Since the network topology changes due to the node mobility, v_i can indirectly send the data packet to v_n using intermediate nodes, v_j or/and v_k. In this paper, for the sake of simplicity [17], we consider upto three-hop relays in the indirect data transfer, because we have witnessed a significant increasing of computing complexity in solving Eq. 3. The expected data delivery latency (E) of single-hop ($T_{i,n}$) and multi-hop ($T_{i,j,n}$ and $T_{i,j,k,n}$) data transfers can be computed in Eqs. 4, 5, and 6 based on Eq. 3, respectively.

$$T_{i,n} = \int_0^\infty \lambda_{i,n} e^{-\lambda_{i,n} x} x \, dx = \frac{1}{\lambda_{i,n}}. \tag{4}$$

$$\begin{aligned} T_{i,j,n} &= \textstyle\int_0^\infty (\frac{\lambda_{j,n}}{\lambda_{j,n} - \lambda_{i,j}} \lambda_{i,j} e^{-\lambda_{i,j} x} + \frac{\lambda_{i,j}}{\lambda_{i,j} - \lambda_{j,n}} \lambda_{j,n} e^{-\lambda_{j,n} x}) x \, dx \\ &= \quad \frac{\lambda_{j,n}}{\lambda_{j,n} - \lambda_{i,j}} \frac{1}{\lambda_{i,j}} + \frac{\lambda_{i,j}}{\lambda_{i,j} - \lambda_{j,n}} \frac{1}{\lambda_{j,n}} = \frac{1}{\lambda_{i,j}} + \frac{1}{\lambda_{j,n}}. \end{aligned} \tag{5}$$

$$T_{i,j,k,n} = \frac{1}{\lambda_{i,j}} + \frac{1}{\lambda_{j,k}} + \frac{1}{\lambda_{k,n}}. \tag{6}$$

Since the data is transferred using the shortest path, the expected data delivery latency between v_i and v_n should be the minimum latency among $T_{i,n}$, $T_{i,j,n}$ and $T_{i,j,k,n}$, e.g., $E_{i,n} = \text{Min}(T_{i,n}, T_{i,j,n}, T_{i,j,k,n})$.

Data Utility Function Since each node has a limited storage space, it will not blindly exchange data with any encountered node. In this paper, we define a data utility function to judiciously decide whether to exchange data. The basic idea is that we consider both time-to-live of data and expected data delivery latency to maximize data access stored in a cache.

The data (m) utility, U_m, can be expressed as,

$$U_m = U\frac{t}{T}e^{t-T-E_{i,n}}, \tag{7}$$

where U is an initial data utility. $E_{i,n}$ denotes the expected data delivery latency from a source (v_i) to a destination (v_n). Both t and T denote the residual and initial time-to-live, respectively. In Eq. 7, the differential form of U_m, $\frac{d_{U_m}}{d_t} = \frac{U}{T}(1+t)e^{t-T-E_{i,n}} > 0$, implies that U_m reduces while t reduces. If $t = 0$, then $U_m = 0$ and the m will be removed from the cache. If $t = T$ and $E_{i,n} = 0$, then U_m has the initial data utility, U. The data utility is always greater than or equal to zero, $U_m \geq 0$, and it is inversely proportional to the expected data delivery latency. For example, if there are k number of nodes that are interested in receiving the m, the data utility can be expressed as,

$$U_m = \sum_{n=1}^{k}(U\frac{t}{T}e^{t-T-E_{i,n}}). \tag{8}$$

In addition, suppose a set of nodes located one-hop apart from v_n (g_n) caches a copy of the m. If v_n can receive the m from its one-hop neighbor nodes, then the data utility of m cached in v_i decreases. Thus, the data utility can finally be revised as,

$$U_m = \sum_{d=1}^{k}(U\frac{t}{T}e^{t-T-E_{i,d}}\prod_{k\in g_n}(1-P_{k,n}(t))), \tag{9}$$

where $P_{k,n}(t)$ is the probability that v_n receives the m from one of its one-hop neighbor nodes (e.g., v_k) during the t. This can also be expressed as,

$$P_{k,n}(t) = \int_0^t \lambda_{k,n}e^{-\lambda_{k,n}x}\,dx = 1 - e^{\lambda_{k,n}t} \tag{10}$$

Data Utility Optimization We further consider how to optimize exchanging data and propose a heuristic approach for cooperative caching between nodes to efficiently access the data directly and indirectly. For example, when two nodes

(e.g., v_a and v_b) encounter each other, they exchange the information of cached data. Note that each node in fact exchanges the ST with any encountered node through the basic pub/sub operations and thus, it knows which node requests what data. After exchanging the ST and DDT, each node computes the data utility of its cached data using Eq. 9. Then each node begins to exchange the data to maximize the benefit of accessing the data based on the following model. Here, suppose v_a and v_b currently cache the k number of data, respectively.

$$\begin{aligned} Max\ Z_1 &= \sum_{i=1}^{k} x_{1,i} u_{a,i} + \sum_{j=1}^{k} y_{1,j} u_{b,j}. \\ Z_2 &= \sum_{i=1}^{k} x_{2,i} u_{a,i} + \sum_{j=1}^{k} y_{2,j} u_{b,j}. \end{aligned} \tag{11}$$

s.t.

$$\sum_{i=1}^{k} (x_{1,i} + x_{2,i}) s_i \leq S_a \qquad \sum_{j=1}^{k} (y_{1,j} + y_{2,j}) s_j \leq S_b \tag{12}$$

$$x_{1,i} + y_{1,j} \leq 1 \qquad (i = j) \tag{13}$$

$$x_{2,i} \leq y_{1,j} \qquad y_{2,j} \leq x_{1,i} \qquad (i = j) \tag{14}$$

$$x_{1,i}, x_{2,i}, y_{1,j}, y_{2,j} \in [0, 1] \quad i, j \in [1, n] \tag{15}$$

Here, a cache size of v_a and v_b is S_a and S_b, respectively. $u_{a,i}$ and $u_{b,j}$ are the data utility of data item i (m_i) and j (m_j) of v_a and v_b, respectively. Decision variables denote the m_i will be ($x_{1,i} = 1$ or $x_{2,i} = 1$) and will not be ($x_{1,i} = 0$ or $x_{2,i} = 0$) stored in v_a, respectively. Likewise, decision variables denotes the m_i will be ($y_{1,i} = 1$ or $y_{2,i} = 1$) and will not be ($y_{1,i} = 0$ or $y_{2,i} = 0$) stored in v_b, respectively.

Eq. 11 can lead to a multi-objective linear programming problem. The first object Z_1 is used to maximize the data utility of all the k number of data cached in v_a and v_b. The second object Z_2 indicates that if there is available space in the cache after the k number of data are allocated, the data that can further maximize the benefit in accessing data will be stored. Eq. 12 indicates that stored data cannot exceed the cache size, where s_i is the size of m_i. Eqs. 13 and 14 indicate that all the k number of data are stored in v_a and v_b, respectively.

Note that Eq. 11 is NP-hard because of its similarity to the 0-1 knapsack problem. Thus, we solve the problem in a heuristic approach. According to the data utility, we first sort all the k number of data cached in v_a and v_b, respectively. Then we select a data item (e.g., m_i) having the largest data utility and store it in the corresponding node. Suppose the m_i is stored in v_b. We mark m_i as it has been processed in v_a. We also select a data item (e.g., m_j) having the second largest data utility and store it in the corresponding node. Suppose the m_j is stored in v_a. We also mark m_j as it has been processed in v_b. We continue to allocate all the data based on the data utility until one of node's cache becomes full. For example, if the cache of v_a is not full, the rest of data will be allocated based on the data utility. If the cache of v_a is still not full, the

Notations:
- $SUT_a[m_{id}, ut, size]$, S_a, and S_b are defined before.
- $TT_a[m_{id}]$: A temporary table in a node v_a, where a list of data packets are stored.
- TS_a: The total size of data packets stored in TT_a.

◇ When v_a and v_b optimize exchanging data to maximize the data utility:

```
while SUT_a ≠ ∅ ∧ SUT_b ≠ ∅
    /* Max returns the id of data packet having the largest ut. */
    id' = Max(SUT_a); id'' = Max(SUT_b);
    if SUT_a[id'].ut ≥ SUT_b[id''].ut
        if TS_a + SUT_a[id'].size ≤ S_a /* Eq. 12 */
            TS_a += STU_a[id'].size; TT_a = TT_a ∪ id';
            Remove m_id' from SUT_a and SUT_b;
        else
            Remove m_id' from SUT_a;
    else
        if TS_b + SUT_b[id''].size ≤ S_b /* Eq. 12 */
            TS_b += STU_b[id''].size; TT_b = TT_b ∪ id'';
            Remove m_id'' from SUT_b and SUT_a;
        else
            Remove m_id'' from SUT_b;
```

◇ When v_a maximizes the data utility based on TT_b:

```
for m_id ∈ TT_b
    Compute ut of SUT_a[m_id];
for m_id ∈ SUT_a
    if TS_a + SUT_a[m_id].size ≤ S_a
        TS_a += STU_a[m_id].size; TT_a = TT_a ∪ m_id;
Store the data packet in TT_a;
```

Fig. 3. The pseudo code of major operations in the PCC.

data that are already allocated to v_b will be allocated based on their data utility to v_a. Thus, v_a and v_b can allocate the data cooperatively and access them directly and indirectly with the minimized number of hops. The major operations of PCC are summarized in Fig. 3.

4 Performance Evaluation

4.1 Testbed Setup

We conduct extensive simulation experiments using the opportunistic network environments simulator (ONE) [4], which is originally developed to evaluate the routing and application protocols in delay tolerant network (DTN) environments. We use an experimental trace, *Infocom 06* [18], collected from realistic environments to evaluate our performance. We set 10 channels in the network, where each node subscribes a channel and generates data from the rest of nine channels in random. The data generation rate follows a uniform distribution. In

Table 1. Simulation Parameters

Parameter	Value
Cache size	1 Mb
Data size	10 to 125 Kb
Data generation rate	0.8 to 12 packets/hour
Data lifetime	8×10^4 to 18×10^4 secs
Number of nodes	98

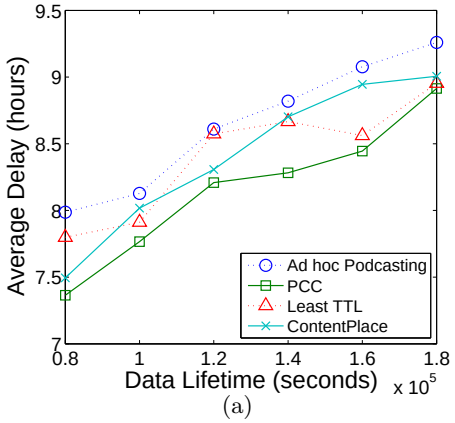
the experiment, we use the first half of the traces for the learning process to acquire the information of nodes' inter-contact time and their subscribed channels. Then the second half of the traces is used to conduct the performance evaluation study. The major simulation parameters are summarized in Table 1.

We evaluate the PCC in terms of three performance metrics by changing data lifetime, data size, and data generation rate: (i) average delay; (ii) average number of delivered data; and (iii) average number of hops. We compare the performance of PCC with three existing schemes:

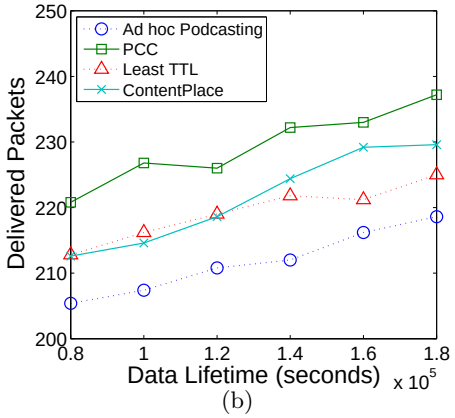
- *Ad hoc Podcasting [2]:* A node first accepts all subscribed data from its neighbor nodes and then randomly accepts the rest of data until its cache becomes full.
- *ContentPlace [3]:* A node first accepts all subscribed data from its neighbor nodes and then accepts the rest of data based on the data utility until its cache becomes full.
- *Least TTL:* A node first accepts all subscribed data from its neighbor nodes and then accepts the rest of data based on time-to-live (TTL) until its cache becomes full.

4.2 Performance Comparison

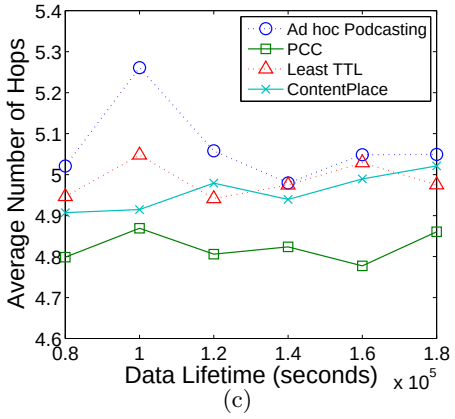
Impact of Different Lifetime We first change the data lifetime to evaluate the performance. Intuitively, more data have a chance to be delivered to subscribed nodes but it could incur the network traffic. In Subfig. 4(a), as the lifetime increases from 8×10^4 to 18×10^4 secs, the average delay for delivering data increases. The longer the lifetime is, the more data is delivered before they are expired. In Subfig. 4(b), the number of delivered data increases with longer data lifetime because more data have a chance to be delivered before they are expired in the network. In Subfig. 4(c), the number of hops does not change much with different lifetimes. As shown in Fig. 4, since the PCC caches the data with high data utility and cooperatively shares them among nodes, it outperforms three other schemes. The PCC tries to maximize the data utility when two nodes are encountered. Although the TTL is an important factor to cache the data, the PCC also considers the freshness of data and data utility. Thus, the performance of the Least TTL is worse than that of the PCC. The Ad hoc Podcasting shows the lowest performance because it does not consider the data



(a)



(b)



(c)

Fig. 4. Performance of data access with different data lifetime. Here, data size and data generation rate are set to 25 Kb and 2 packets/hour, respectively.

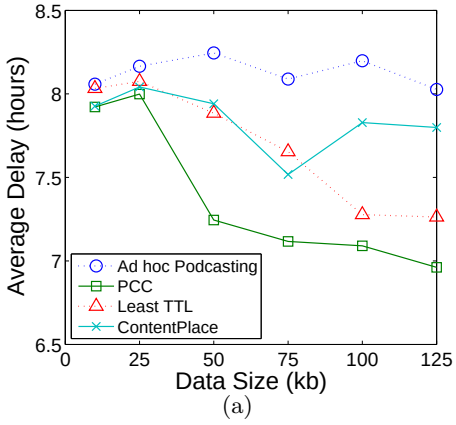
utility but randomly accepts data to store. In the ContentPlace, each node caches the data with high data utility but does not consider to optimize the benefit in accessing data with other nodes.

Impact of Data Size Second, we change the data size to evaluate the performance. In Subfig. 5(a), the delay decreases as the data size increases. Based on the data freshness, older data are discarded and this can contribute to reduce the delay. In Subfig. 5(b), as the data size increases, the number of delivered data decreases. Note that when the data size is 125 Kb, each node can maximumly store eight data and thus, the number of delivered data becomes small. In Subfig. 5(c), the number of hops is between 4 to 5.5 and it shows relatively stable with different data sizes. As shown in Fig. 5, the PCC outperforms three other schemes. The bigger the data size is, the better PCC performs. This is because when the data size is small, nodes can cache all the received data. When the data size increases, however, nodes selectively cache data. The PCC efficiently exchanges the data and optimize the benefit in accessing data from other nodes indirectly and thus, it can achieve better performance.

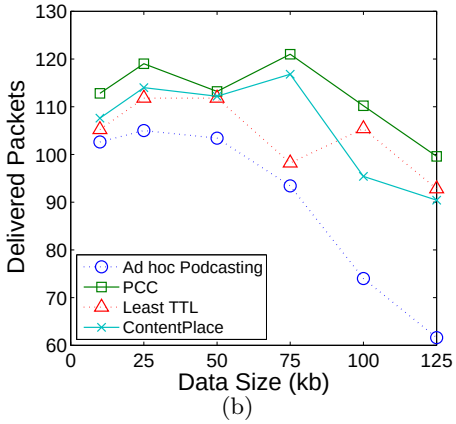
Impact of Data Generation Rate Third, we change the data generation rate to evaluate the performance. In Subfig. 6(a), the delay decreases as the data generation rate decreases because each node has a limited cache size. As the rate increases, nodes tend to drop some cached data because of newly generated data. When the rate is 0.8 packets/hour, nodes do not drop any new data generated and thus, this can lead to the highest delay. In Subfig. 6(b), the number of delivered data decreases as the rate decreases. In Subfig. 6(c), overall number of hops are between 4.5 to 6. Similar to the results as shown in Figs. 4 and 5, the PCC achieves better performance than that of other three schemes. The Ad hoc Podcasting performs worst because nodes randomly store the data without considering any data utility. Although both Least TTL and ContentPlace show better performance than that of the Ad hoc Podcasting, they only consider the locally optimized data utility for caching without considering the data cached in other nodes.

5 Conclusion

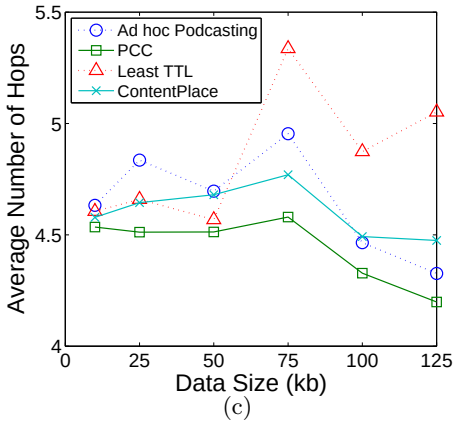
In this paper, we proposed a probabilistic cooperative caching-based content subscribing technique, called PCC, to achieve efficient data access in the pub/sub based OppNets. We realized the PCC by measuring the expected data delivery latency, building a data utility function, and conducting the data utility optimization. The PCC considers the inter-contact time, TTL, and other nodes' access interests. The PCC efficiently exchanges the data to maximize the data utility and optimizes the benefit in accessing data directly and indirectly. Extensive simulation results showed that the PCC outperforms other three existing schemes in terms of the delay, number of delivered data, and number of hops.



(a)

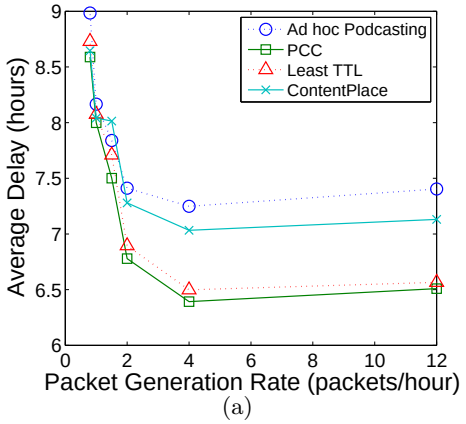


(b)

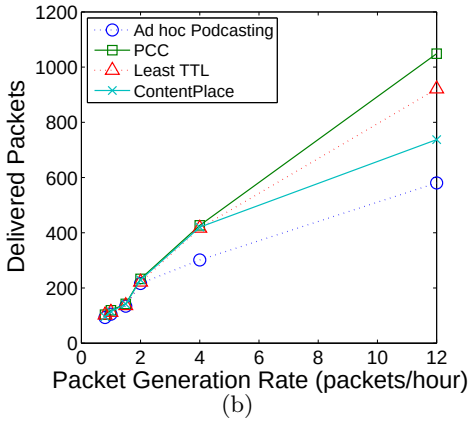


(c)

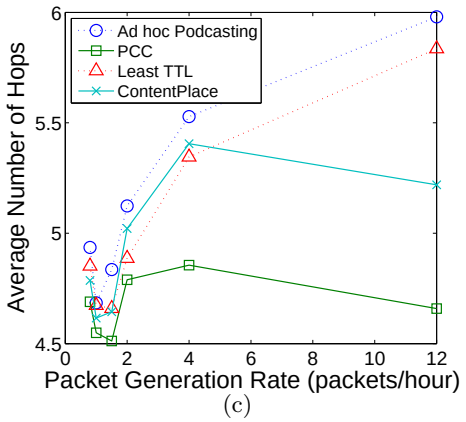
Fig. 5. Performance of data access with different data size. Here, data generation rate and lifetime are set to 1 packet/hour and 18×10^4 secs, respectively.



(a)



(b)



(c)

Fig. 6. Performance of data access with different packet generation rate. Here, data size and lifetime are set to 25 Kb and 16×10^4 secs, respectively.

To see the full potential of our scheme, we are currently extending the basic sub/pub operations and cooperative caching in terms of cache admission control and replacement policy to enhance data availability and accessibility in the PCC.

Acknowledgment

This work is supported in part by China Scholarship Council and GuangDong University of Foreign Studies (15T26).

References

1. W. Rao, K. Zhao, Y. Zhang, P. Hui, and S. Tarkoma, "Towards Maximizing Timely Content Delivery in Delay Tolerant Networks," *IEEE Trans. on Mobile Computing*, vol. 14, no. 4, pp. 755–765, 2015.
2. M. May, V. Lenders, G. Karlsson, and C. Wacha, "Wireless Opportunistic Podcasting: Implementation and Design Tradeoffs," in *Proc. ACM CHANTS*, 2007, pp. 75–82.
3. C. Boldrini, M. Conti, and A. Passarella, "ContentPlace: social-aware data dissemination in opportunistic networks," in *Proc. ACM MSWiM*, 2008, pp. 203–210.
4. A. Keranen, J. Ott, and T. Karkkainen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN)*, 2013.
5. A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," Duke University, Tech. Rep. CS-200006, May 2000.
6. T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in *Proc. ACM SIGCOMM WDTN*, 2005, pp. 252–259.
7. J. Leguay, T. Friedman, and V. Conan, "DTN Routing in a Mobility Pattern Space," in *Proc. ACM SIGCOMM WDTN*, 2005, pp. 276–283.
8. H. Zhou, J. Wu, H. Y. Zhao, S. J. Tang, C. F. Chen, and J. M. Chen, "Incentive-Driven and Freshness-Aware Content Dissemination in Selfish Opportunistic Mobile Networks," in *Proc. IEEE MASS*, 2013, pp. 333–341.
9. E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks," in *Proc. ACM MSWiM*, 2007, pp. 225–234.
10. A. Socievole, E. Yoneki, F. D. Rango, and J. Crowcroft, "ML-SOR: Message routing using multi-layer social networks in opportunistic communications," *Computer Networks*, vol. 81, no. 4, pp. 201–219, 2005.
11. V. Lenders, G. Karlsson, and M. May, "Wireless Ad Hoc Podcasting," in *Proc. IEEE SECON*, 2007, pp. 273–283.
12. F. Li and J. Wu, "MOPS: Providing Content-Based Service in Disruption-Tolerant Networks," in *Proc. IEEE ICDCS*, 2009, pp. 526–533.
13. L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE Trans. on Mobile Computing*, vol. 5, no. 1, pp. 77–89, 2006.
14. E. Nordstrom, C. Rohner, and P. Gunningberg, "Haggle: Opportunistic mobile content sharing using search," *Computer Communications*, vol. 48, no. 15, pp. 121–132, 2014.

15. Y. S. Wang, J. Wu, and M. J. Xiao, "Hierarchical cooperative caching in mobile opportunistic social networks," in *Proc. IEEE GLOBECOM*, 2014, pp. 411–416.
16. H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni, "Recognizing Exponential Inter-Contact Time in VANETs," in *Proc. IEEE INFOCOM*, 2010, pp. 1–5.
17. W. Gao, G. Cao, A. Iyengar, and M. Srivasta, "Supporting Cooperative Caching in Disruption Tolerant Networks," in *Proc. IEEE ICDCS*, 2011, pp. 151–161.
18. J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "Crawdad data set cambridge/haggle (v. 2009-05-29)."