ACC: Anonymous Cooperative Caching in Wireless Ad Hoc Networks

Zhaoyan Su\textsuperscript{1}, Xiapu Luo\textsuperscript{2}, Weigang Wu\textsuperscript{1}, Jiannong Cao\textsuperscript{2}
1. Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China
2. Department of Computing, The Hong Kong Polytechnic University, Hong Kong
\{suzhyan,wuweig\}@mail.sysu.edu.cn, \{csxluo, csjcao\}@comp.polyu.edu.hk

Abstract—Cooperative caching is an effective approach to share cache copies among network nodes and then reduce data access cost in ad hoc networks. Since cached data may be privacy sensitive, privacy preservation should be considered. In this paper, we study this problem by achieving anonymity in cooperative caching. Our anonymous caching protocol borrows idea from anonymous routing protocols. Data query and access are realized based on anonymous hop-by-hop broadcasting. Pseudonyms are used in communications so as to protect the real identity. Security analysis confirms that our protocol can achieve sender anonymity, receiver anonymity, and communication anonymity. We also conduct simulations to evaluate the effect of anonymity mechanism on system performance. Different variants of anonymous caching algorithm and non-anonymous algorithms are simulated for comparisons. The results show that, anonymity will cause additional message cost and time cost, but the cost is reasonable. To the best of our knowledge, this is the first work on anonymity in cooperative caching.

Keywords—Data cache; cooperative caching; anonymity; privacy; ad hoc network.

I. INTRODUCTION

Due to the advantages in flexible deployment, low cost and easy maintenance, wireless ad hoc networks are especially suitable for the scenarios where the deployment of network infrastructure is too costly or even impossible [5].

Cooperative caching (CC) is an effective way to reduce data access cost in ad hoc networks. Besides the cache copy at local memory/storage, CC allows a node to access the cache copy at other nodes, i.e., cache copies are shared by the whole network. This will improve cache utility and reduce data access cost, including both communication cost and time cost. Moreover, when data are delivered through multi-hop paths, intermediate nodes can cache the data forwarded for future use. This can further reduce the data access cost.

CC involves three key issues: cache placement [20], cache discovery [3][22], and cache consistency [11][14]. Cache placement refers to determining where and what to cache; cache discovery refers to the mechanism to find and obtain a cached data item; and cache consistency means to ensure that the data value in cache copies is consistent with the source copy at the data server. The first two problems are so closely related that they are usually studied together.

Cooperative caching, especially cache placement, has been studied by many researchers. The optimal cache placement in ad hoc networks has been proved to be NP-hard, even if only one data item is considered [1][20]. Existing works on cache placement mainly focus on how to make use of the data access frequency information and network topology information in selecting cache nodes [10][12][20][25][28][30][32].

For cache discovery, many efforts have been made on combining passive and active query approaches [3][4][6][7][22]. In the passive approach [29][30], the cache copy at one node is unknown to other nodes and it is used if a request reaches a cache node. In active cache discovery, a node obtains the knowledge of the cache node first and then the request is sent to the cache node rather than the source node. Active cache discovery can be further divided into two categories: notification based and query based. In the former, a cache node needs to inform other nodes that it has a cache copy [25] [27], while in the reactive approach [22], a node needs to locate the cache node by query before sending out its request.

All the existing works on cooperative caching, however, do not consider the privacy problem in cache sharing. Privacy is attracting more and more attention from both academia and industry. Although a plethora of anonymity protocols and systems have been proposed [9] [15][21], almost all of them focus on anonymous routing. Cooperative caching is a quite different from routing. In routing protocols, the destination is known by the source and the objective to find the path to reach the destination. In cooperative caching, there may be no or several cache nodes (destination) that can provide cache copy wanted and the key point of cache discovery is to find out cache nodes and determine which cache node should be contacted to get data. Cooperative caching with preserving privacy is a challenging task. Cooperative caching requires network nodes to share their data access information and data items cached, which will disclose privacy information to others.

In this paper, we propose the concept of anonymous cooperative caching (ACC) and design protocols to realize it. ACC requires that both the requestor and provider of a cache copy are anonymous to the network. That is, a node is able to access the cache copy at other nodes, but no others can know the real identification of the node that requests the data and the node provides the cache. Even the requestor and provider do not know the real identification of each other. Moreover, the data item should be invisible to intermediate nodes that forwarding the request or reply.

Existing cache protocols/algorithm rely on the network topology information and data request information to determine which data should be cached and where to find the cache copy of data requested. In ACC, however, network topology cannot be known by a node, and data request/reply should be encrypted. Therefore, to realize ACC, both cache discovery and cache placement need to be reconsidered and new protocols must be designed.

978-0-7695-5104-3/13 $26.00 © 2013 IEEE
DOI 10.1109/MASS.2013.74
Our proposed protocol realizes anonymous cache discovery base on anonymous broadcasting. When a node wants to access some data, it first queries other nodes about whether a cache copy is available. To achieve anonymity, only pseudonym is used in the query packet. With the propagation of query packet, cache nodes will be reached and anonymous path between the requestor and cache node is established via the response from the cache node. We borrow idea from anonymous routing protocols [2][16] to establish the anonymous path. However, cache discovery has fundamental difference from anonymous routing, which is discussed in the next section.

Anonymous cache placement is realized by a pushing based mechanism. With anonymity, accurate and efficient cache placement is difficult. On one hand, it is difficult for an intermediate node to cache the data copy being forwarded. On the other hand, the data request information passing by is also invisible for an intermediate node, data access frequency is not as accurate as in non-anonymous scenarios. We adopt push-based mechanism to let the data source proactively push the data to other nodes so as to improve the availability of data copies to cache.

We then analyze the security of our protocol. It can defend against most of possible attacks usually considered in privacy preservation. We have also conducted extensive simulations to evaluate the performance of our proposed algorithm, in terms of communication cost and time delay. The simulation results show that, there is a tradeoff between anonymity and efficiency. Compared with non-anonymous protocol, the cost of anonymity is reasonable and acceptable.

The rest of the paper is organized as follows. Section II firstly presents the system model and then defines the ACC problem. The proposed ACC protocol is described in Section III. We analyze the security of our proposed protocol in Section IV, including both anonymity analysis and attack analysis. Performance evaluation via simulations is reported in Section V. Finally, Section VI concludes the paper with future directions.

II. SYSTEM MODEL AND PROBLEM FORMALIZATION

A. System Model

We consider a wireless ad hoc network that consists of \( n \) nodes, which communicate by sending and receiving messages through wireless links. Two nodes can communicate directly if they are in the communication range of each other. For the purpose of privacy preservation, the nodes do not know each other and they can only communicate via broadcasting messages anonymously.

There are multiple data items to be accessed by network nodes. Each data item is served by its source node. One source node may serve multiple data items. The data source is known to all the nodes. The knowledge of data source is obtained by some way outside the caching system.

To reduce data access cost, each node caches data items subject to its memory capacity limitation. The node with a cache copy of some data item is called a cache node of that data item. For simplicity of presentation, the data source is also viewed as a cache node.

To improve the utility of cache copies, the nodes would share their caches with others. That is, if a request reaches a node and it has cached a copy of the data requested, it will reply the request by sending data copy even if the node is not the target node of the request. However, to preserve anonymity, neither the requestor nor the replier will release their real identification.

Both symmetric and asymmetric encryptions are involved in anonymity preservation. We assume RSA and AES are used for asymmetric encryption and symmetric encryption respectively.

B. The Anonymous Cooperative Caching Problem

We assume that the attacker can passively eavesdrop on all the radio transmissions and can insert a few malicious nodes that cache some sensitive data items. The goal of an attacker is two-fold. First, when monitoring a user, the attacker wants to know which data item she is requesting and from which node she fetches such data item. Second, when behaving as a cache, the attacker wants to infer who downloads the sensitive data item.

We do not consider the attacks based on traffic analysis [19] or traffic watermarking [13], because they can be defeated by link padding [26] or be detected [18]. Other common security issues in ad hoc networks, such as black hole attack, DoS, etc., are beyond the scope of this paper. To prevent being identified through physical address like MAC address by the attacker, we assume that all nodes turn on the promiscuous mode of their network interfaces to receive wireless frames so that there is no need to put the physical address in the wireless frames.

Our protocol intends to achieve three targets:

1) Requestor anonymity. The requestor of a data item should be anonymous to the network. That is, none, except the requestor itself, can know the real identification of the requestor.

2) Provider anonymity. The provider of a data copy should be anonymous to the network. That is, the real identification of the node that provides data copy should not be disclosed to others.

3) Communication anonymity. The attacker cannot tell whether and when a requestor and a provider communicate.

III. DESIGN OF THE ACC PROTOCOL

A. Overview of the Protocol

The protocol consists of four steps: query, response, request, and reply.

When a node wants to access some data item, it will query other nodes to obtain the information of cache copies. A query message will be broadcast hop-by-hop in ad hoc network. With pseudonyms included in the query message, cache nodes of the requested data will respond via reverse path and the paths (chain of pseudonyms) between requestor and cache nodes are established.

Then, the requestor will choose one cache copy to access by choosing (the first) one path from the paths to cache nodes, and send request message via the path chosen. The cache node that receives the request will reply by sending
data copy back to the requestor. Obviously, the request message and reply both are via the anonymous path established in the query/response phase.

The procedure above is similar to that of routing protocols, where route is discovered via broadcasting and then data packet is sent along the route.

However, due to different requirements of caching system, the procedure above is realized with the following key mechanisms delicately designed for caching.

1) Probability based query broadcasting. Cache discovery is data item oriented and cache placement at the nodes may change from time to time, so cache query is necessary for each new request. To avoid the high cost of broadcasting, we adopt a probability based broadcasting mechanism, which is similar to gossip approaches.

2) Anonymous data pushing. With encryption, the data copy in reply step is not visible to intermediate nodes along the forwarding path. This obviously prevent these nodes from obtaining the data copy being forwarded and reduce the efficiency of caching system. To cope with this problem, we design a mechanism to push data from source to other nodes anonymously in a hop-by-hop way.

B. Data Structures and Packet Types
The key data structure maintained by a node I is the forwarding table. Each record is for the path of one specific data access request. The format of the record is as follows. The following record is used for query/response phase:

\[ \text{SEQ}, \text{DID}, K_I, N_I, \text{timer}, N_{\text{prev}} \]

where,

- \text{SEQ}: the sequence number of the data query, used to distinguish different access requests.
- \text{DID}: the id of the data item wanted.
- \text{N}_I: the pseudonym of I used for the path. It may be different for different record.
- \text{timer}: a timeout that specifies the lifetime of the record.
- \text{K}_I: the symmetric session key that will be used to encrypt the data and route onion.
- \text{N}_{\text{prev}}: the pseudonym of the previous hop.

Totally five different packets are used in our design, which are described as below:

- \text{QRY}<\text{SEQ}, \text{DID}, \text{N}_I, \text{PK}_I, \text{N}_{\text{prev}}>: the query packet to query other nodes for cache copy. Five fields stand for sequence number, data id, pseudonym of data requestor, a public key, pseudonym of previous hop (i.e. sender of the packet), respectively.
- \text{RSP}<\text{SEQ}, \text{N}_{\text{prev}}, \text{DID}, \text{PK}_{\text{temp}}, \text{PK}_b, \text{flg}, \text{TR}, \text{PDO}>: the packet that responds to a query.
- \text{REQ}<\text{N}_I, \text{PRO}> : the request packet to ask a cache node to send data back.
- \text{REP}<\text{N}_I, \text{OR}>: the reply packet from a cache node that carries data back to requestor node.
- \text{PSH}<\text{DID}, \text{DataContent}>: the push packet to push data from a cache node to its neighbors.

C. Detailed Description
As mentioned above, the data access operations of our protocol can be divided into four phases. We describe them one by one. Finally, we present the anonymous pushing mechanism to further reduce data access cost.

1) The query phase
When a node A wants to access some data item \text{DID}, it first creates a pseudonym \text{N}_A for this access, and generates a pair of PKI keys correspondingly. Then, it broadcasts a query packet to its neighbor. The query packet is in the format: \text{QRY}<\text{SEQ}, \text{DID}, \text{N}_A, \text{PK}_A, \text{N}_{A2}> where \text{SEQ} is the sequence number of the query, \text{PK}_A is the public key, \text{N}_{A2} is the pre-hop name generated by Node A to make it different from \text{N}_A. Node A then waits for response from cache nodes.

When a node \text{B} receives the query packet broadcasted by node \text{A} (originated from \text{A}), \text{B} will first check if it has the data wanted.

If \text{C} does not have the data item, \text{C} needs to add a record for the new query: \text{QRY}<\text{SEQ}, \text{DID}, \text{N}_C, \text{timer}, \text{N}_B>, where \text{N}_C is the pseudonym of \text{C} itself and \text{N}_B is the pseudonym of the previous hop \text{B}. Then, \text{C} updates the packet to be \text{QRY}<\text{SEQ}, \text{DID}, \text{N}_A, \text{PK}_A, \text{N}_C, \text{N}_{A2}> and further forwards the query packet to the next hop with a predefined probability \text{p}. If \text{C} has the data, the operation switches to the response phase.

![Figure 1. Example of query phase](image)

2) The response phase
When a cache node \text{D} of the requested data item receives query packet, it will respond to the query to inform the requestor that the data copy is found.

Node \text{D} first generates a temporary asymmetric key pair \text{PK}_{\text{temp}}, \text{SK}_{\text{temp}}, a temporary symmetric key \text{K}_D and a pseudonym \text{N}_D. Then, \text{D} will prepare the trap door \text{TR} and path chain \text{PDO}_D.

\[
\text{TR}=\text{E}_{\text{PK}_D}(\text{N}_D, \text{SK}_{\text{temp}})
\]

which is used to determine if the receiver of the response is the original data requestor. \text{PDO}_D=\{\text{E}_{\text{PK}_D}(\text{K}_D), \text{E}_{\text{K}_D}(\text{N}_D, \text{N}_A)\}, which is an encrypted path chain of pseudonyms. The public key \text{PK} used in \text{PDO} may be either \text{PK}_A the public key of requestor node or \text{PK}_{\text{temp}} the temporary public key generated by the cache node.

Node \text{D} will make a record of the query for future use of data request/reply. The record should be \text{QRY}<\text{SEQ}, \text{DID}, \text{K}_D, \text{N}_D, \text{timer}, \text{N}_B>, where \text{N}_B is the previous hop that has sent query packet to \text{D}.

Finally, \text{D} broadcasts a response packet as \text{RSP}<\text{SEQ}, \text{N}_B, \text{DID}, \text{PK}_{\text{temp}}, \text{PK}_A, \text{flg}, \text{TR}, \text{PDO}_D>, where \text{flg} is a flag to indicate which key is used for encrypting \text{PDO}.

When the intermediate node \text{B} (not the original requestor \text{A}) receives a response packet \text{RSP}<\text{SEQ}, \text{N}_B, \text{DID}, \text{PK}_{\text{temp}}, \text{PK}_A, \text{flg}, \text{TR}, \text{PDO}_D>, it will first look up its forwarding table. If it cannot find the record for pseudonym \text{N}_B, the
node will try to open the trap door

\[ E_K(N_{A}) = \text{PK} \]

\[ \text{either which} = 0 \text{ or which} = 1 \]

node cannot get data from cache nodes and it will send request to

node A and get the chain of pseudonyms to cache node.

Assuming A chooses the path \{(K_{n}, N_{0}), ..., (K_{i}, N_{i}), (K_{D}, N_{D})\}. Node A will prepare an onion packet:

\[ \text{PRO}_{A} = \text{PK}(\text{REQ}(N_{n_{1}}, \text{PRO}), \text{PK}(N_{D}, \text{PRO})) \]

\[ = \text{PK}(\text{REP}(N_{i}, \text{END}, \text{RT})) \]

where, END is a tag to indicate the destination node, and RT is the path chosen by A, i.e. \{N_{i}, K_{i}, ..., N_{n_{1}}, K_{n_{1}}, N_{D}, K_{D}\}. Then, A will broadcast the request packet: \( \text{REQ} < N_{n_{1}}, \text{PRO} > \)

When an intermediate node i receives the request packet \( \text{REQ} < N_{n_{1}}, \text{PRO} > \), it will discard the packet if its forwarding table does not have the record of \( < N_{n_{1}}, K_{i} > \). Otherwise, it

will needs to further forward the packet. Then, node i will decrypt \( \text{PRO}_{i} \) get \( < N_{i}, \text{PRO}_{i} > \), and broadcast \( \text{REQ} < N_{i}, \text{PRO}_{i} > \)

Eventually, node D will receive the request packet \( \text{REQ} < N_{D}, \text{PRO}_{D} > \) and get \( < \text{END}, \text{RT} > \) after decrypting \( \text{PRO}_{D} \).

4) The reply phase

With the request packet \( \text{REQ} < N_{D}, \text{PRO}_{D} > \), cache node D
can get the path chain for data forwarding, i.e. \{N_{D}, K_{D}, ..., N_{n_{1}}, K_{n_{1}}, N_{D}, K_{D}\}. If it does not have the cache data due
to cache replacement or other reasons, D will not send reply and
the requestor will choose other target after timeout. If D
has the data, it will prepare the reply packet:

\[ \text{REP} < N_{i}, \text{OD} > \]

where,

\[ \text{OD} = \text{PK}(\text{END}, \text{DATA}) \]

The following of reply forwarding is similar to that of request forwarding, i.e. each intermediate node will decrypt one layer of the onion packet and forward it towards A.

Finally, node A will get the reply packet \( \text{REP} < N_{D}, \text{END}, \text{DATA} > \). It can then decrypt the packet using its
key \( K_{A} \) and obtain the data requested.

Besides using the data as application required, node A will also determine whether to cache the data for future use, i.e. cache placement will be conducted. What metric can be used to evaluate the significance of data items? Since

distance among nodes is not available in anonymous environment, only popularity based metrics are feasible.

5) Data pushing

In existing cooperative caching systems, an intermediate node can cache data forwarded for other nodes, which is an important way to obtain data. However, with anonymous
design, the data copy carried by a reply packet is not visible
to intermediate nodes, so a node can only get data from
request originated by the node itself, which will reduce the possibility of caching a data.

To address this problem, we propose to let nodes with a copy push a data item to other nodes explicitly so as to help intermediate nodes get more data. To preserve anonymity, the pushing is also done via anonymous broadcasting. That is, a cache node proactively push data copies to its neighbors via broadcast, and the receivers will determine whether or not to cache the data items pushed, based on cache placement metric. It may further push the data to its own neighbors, depending on the specified percentage of hot data.

Since pushing itself will cause additional communication cost, pushing all data items may not be beneficial. A node can push only hot data items, i.e. those frequently requested. The percentage of hot data items can be predefined or dynamically changed, according to optimization strategies.

In the performance evaluation part, we will vary the percentage of hot data to push and discuss its effect.

IV. Security Analysis

We first discuss how anonymity is achieved by our algorithm and then analyze the security against popular attacks in ad hoc networks.

A. Anonymity analysis

Our protocol can achieve sender (the node provide data copy) anonymity, receiver (requestor of the data) anonymity, and communication anonymity.

The real identity of a node in our design is protected through pseudonyms. During the procedure of data access, the request node generates a new pseudonym $N_A$ and a temporary asymmetric public key for a new request. All messages are propagated via broadcasting. Obviously, no other nodes can know the real identity of $N_A$ except the request node itself. An intermediate node can know its previous hop’s pseudonym. It may also know the next hop’s pseudonym by decrypting the onion packet. However, it does not have the information of the entire route. Therefore, receiver anonymity is persevered.

A cache node hides its pseudonym in the onion packet and acts like the intermediate nodes, and the destination node’s pseudonym can only be revealed by the request node which has the private key to decrypt the packet. Therefore, sender anonymity is persevered.

Since communication anonymity is guaranteed as long as either or both of sender anonymity and receiver anonymity are fulfilled. Therefore, our system also provides communication anonymity.

B. Attack analysis

In the following, we discuss defense against six commonly used attacks.

1) Controlling intermediate nodes

An intruder may control one or more intermediate nodes along a route of message propagation. By controlling one intermediate node, an intruder can only know the pseudonyms of adjacent node. With such pseudonyms, the intruder cannot relate the node’s real identity.

Moreover, the requestor can avoid tracking of end node via REP packet by broadcasting the REP packet although it is the destined receiver.

An intruder cannot get the complete route info unless it controls all the nodes on the route, including the request node and the cache node because the intruder needs their secret key to decrypt packets (intermediate nodes use the requestor’s public key and cache node’s public key alternately). However, data access then becomes “internal” operations among comprised nodes, which is not threatening.

2) Iterated compromise

In such an attack, if one node is compromised, the intruder can obtain the pseudonym of the previous hop via forwarding table and accordingly can compromise the previous hop. Iteratively, the intruder can compromise the requestor node, which is unacceptable. However, such attack can only be completed after quite a long time since intruding neighbor node is time consuming. Considering that the entries in forwarding table are kept for a specified time period, iterated compromise cannot be completed if a short timeout for forwarding entry is set. We can also use hash functions to protect some field of the forwarding table.

3) Distance information

In the end of request phase, the requestor will get all pseudonyms on the route. To avoid distance analysis, an intermediate node can add a loop to itself randomly.

4) Denial of Service (DoS)

In DoS attacks, the malicious nodes can keep broadcasting request packets, which will exhaust both computation and network resources, and the network may be paralyzed. What’s worse, the anonymous mechanism makes it difficult to trace the attacker because it is legal for an attacker to use different pseudonyms to broadcast requests. Such DoS attacks are hard to defense, as in other environments.

5) Fake data

Currently, fake data is possible since no mechanism is considered to guarantee that the received data copy is demanded by the request node. However, such an attack is can be defeated via data validation. For example, the request node can use MD5 to generate the message digest and add it to the “If-None-Match” field of HTTP header and send it back to the data server to check the data. The server can then check whether the data has been modified and send feedback to the request node.

6) Timing analysis

In timing analysis attack, a malicious node can get global knowledge of packets’ transmit time. Then, it can trace those packets with the same sequence number and guess their transmission sequence according to transmit time. This may result in the reveal of the route information. To defense such attacks, the cache node can further transmit a QRY packet randomly with some probability so as to add noisy packet information and conceal the packet sequence.

V. Performance Evaluation

To evaluate the performance of our proposed protocol, we have carried out extensive simulations using the ns-3 simulator. We first examine the effect of pushing. Different
percentages of popular data, and then simulate different protocols for the purpose of comparison, including three variants of our protocol, and one non-anonymous protocols.

The first variant is the complete version of our protocol as described in Section IV. The second is a simplified version with push mechanism removed. That is, a node can get data to cache only from its own requests. The third one differs from the second version in that, the data copy is not encrypted, so that intermediate node can cache it when the data passing by.

The last algorithm is non-anonymous at all. Since no anonymity is preserved, real id and routing protocol can be used for data forwarding. Same as our algorithm, four phases are carried out to get data. However, no encryption and pseudonym are needed. All the nodes communicate using their real ids and OLSR is employed to establish paths among any pair of nodes. After the cache node id is obtained via query-response, the request and reply messages are routed via OLSR. Intermediate nodes can then easily obtain request information of other nodes and cache data passing by.

The results of four protocols are tagged with Anony-full, Anony-simp, Anony-plaindata, Non-anony respectively.

The key settings are listed in Table I.

## A. Simulation Setup and Performance Metric

Basically, we follow the setup used in [25][28]. All the key settings are listed in Table I.

### 1) Network Setup

We simulated a wireless ad hoc network with 25 nodes, which are uniformly distributed in a territory of 500×500 m². For the MAC layer protocol, IEEE 802.11 was used. The transmission range is about 150 meters. We use OLSR as the routing protocol because it is popular and included in transmission range. We use OLSR as the routing protocol because it is popular and included in transmission range. The MAC Protocol is IEEE 802.11b.

### 2) Data Access Model

All the network nodes, including the sources, act as client nodes. Following [25][28], each client sends out data access requests with a constant rate. The time interval between two consecutive requests follows the exponential distribution with the mean value varied from 5 to 40 seconds. The timeout for a reply is set to 40 seconds. To get stable results, each simulation was executed for 10000 seconds.

Two data access patterns have been simulated. The first one is simply random pattern, where each node each node uniformly accesses 200 randomly chosen data items. The second pattern is location-based, where two neighbor nodes have common candidate data items. More precisely, each node has 200 candidate items and 30% of them are also the candidates of its left or right neighbor.

### 3) Performance metric

We measured five different metrics to demonstrate the performance of the data caching system.

- **Cache Hit Ratio (HR):** the percentage of successful requests replied by cache nodes rather than the data source.
- **Access Path Length (PL):** the average length of the path between a data requestor and the corresponding replier. This metric directly show how far a data copy is.
- **Message Cost per request (MC):** the average number of packet transmissions done for each successful data access. This includes all types of packets, including QRY, RSP, REQ, REP, and PSH. This metric directly reflects the efficiency of a caching system in terms of communication cost.
- **Access Delay (AD):** the average time interval between sending a data request and receiving the corresponding reply. This is obvious the metric for time cost.
- **Success Ratio (SR):** the percentage of the requests that are replied before the timeout occurs.

### B. Examination of hot data percentage

The percentage of hot data, i.e. data to be push is a key parameter affecting the performance of the whole cache system. To examine such effect we simulate the full anonymous algorithm with different percentages of hot data, from 0% to 100%. The results are plotted in Fig. 5.

HR varies between about 65% to 80% under various hot data percentages. This indicates that cooperative cache copies play a significant role in data access and cooperative caching is effective. The value of hot data percentage affect cache hit ratio significantly. HR increases with the increase of hot data percentage and reach the peak value when 70% of data items will be push to neighbors. This is expected because more data being push will add more cache copies. However, HR decreases a little when the hot data percentage becomes too large. This may be caused by packet loss due to high link contention if too many data items are push. This can also explain the change of access delay.

The change of PL under various hot data percentage is simpler. It increases first and then become stable. This is because, if the cache space has been filled completely, more pushing will not increase the total number of caches.

MC decreases first and then increases. The overall message cost includes two parts: message cost of pushing data and message cost of accessing data. With the increase of hot data percentage, the cost of pushing data should increases...
accordingly. However, the change of data access cost will decreases when hot data percentage is small. When the hot data percentage becomes large, the decrease of data access cost becomes very slow, which is indicated by the change PL. Then, as the sum of pushing message cost and data access cost, MC decreases first and then increases.

Now, let us examine the results of SR. A request may fail in two cases, loss of messages or too long delay. The effect of hot data percentage on message delay is shown in Fig. 5-(c). Packet loss should roughly vary inversely as the message cost. Combining Fig. 5-(c) and Fig. 5-(d), we can easily understand the change of SR in Fig. 5-(e).

now, let us compare different algorithms. In all cases, the simple anonymous algorithm is the worst. Non-anonymous algorithm outperforms the anonymous algorithm with plain data although the difference is not large. This indicates that anonymity itself will reduce the efficiency of caching, due to more difficult to obtain data request (frequency) information and data copy to cache. Thanks to the pushing mechanism, the full anonymous algorithm can achieve higher HR than non-anonymous one in all cases.

2) Access Path Length (PL)

Fig. 7 shows the results of PL against request arrival rate. Generally, the faster the requests arrive, the fewer hops are needed to access data. As discussed in the part of HR, when a data item is accessed, it may be cached by the requestor and other nodes forwarding the reply. Then, if some node requests the same data after a short time, with a high probability, the cache copy is still there. Therefore, PL decreases with the increase of request arrival rate. Another factor, cache space size also affects PL obviously. With a larger cache space, more data copies can be cached and PL becomes smaller.

The effect of access pattern is shown by comparing Fig. 7-(a)/(c) with Fig. 7-(b)/(d). PL under location-based pattern is smaller than that under random pattern. This is easy to understand. In the location based pattern, the nodes close to each other have more common interests in data access, so cache nodes are also close to the requestor and consequently the access path length is reduced.

Now, let us compare the four algorithms. The algorithm with simple anonymity performs the worst, which is easy to understand. Without caching at intermediate nodes, few copies are available and consequently, a request can only be met with a node far away. However, even the anonymous algorithm with plain data, where intermediate nodes can cache data passing by, is worse than the one without anonymity. This can be explained as follows. Without anonymity, node ids are public and visible to all others. The request message and reply message are forwarded along path established by OLSR, which can find better and shorter path than the one established by anonymous broadcasting.
It is interesting to see that the full anonymous algorithm achieves the smallest access path length. This clearly shows that the push mechanism can significantly increase cache copies in the system.

3) Access Delay (AD)

Results of access delay are shown in Fig. 8. The effect of request arrival rate on AD is quite different from that on PL. Under a high request arrival rate, the traffic of the network is high, so the message delay is large and as a result, the data access delay increased. Data access pattern and cache space size have effect very similar to that on PL.

![Figure 7. Results of access path length](image)

Comparing the four algorithms, we can see that non-anonymous algorithm can get data much faster than the other three. This is because of two reasons. Firstly, the encryption operations in anonymous algorithms will certainly introduce additional time cost. Secondly, the packets (except of QRY) in non-anonymous protocol are forwarded along route established by routing protocol, which is more efficient and faster than hop-by-hop broadcast with pseudonyms.

The difference between the three anonymous algorithms is similar to that in PL. The full anonymous algorithm performs better than the other two in various cases. This indicates that the PL determines which one is faster because a smaller PL means fewer hops and then shorter access delay.

4) Message Cost (MC)

Fig. 9 shows the results of overall message cost. The effect of request arrival rate on MC is similar as on PL. This is easy to understand, because, a small PL will certainly reduce the messages cost.

The difference among algorithms in MC is similar to that in AD. Non-anonymous algorithms has the smallest message cost mainly because the efficient message forwarding via OLSR in request/reply.

![Figure 9. Results of message cost](image)

5) Success Ratio (SR)

We plot the result of success ratio in Fig. 10. Generally, all the algorithms achieve very high success ratio. It is interesting to see that effect of request arrival rate is different under different cache space sizes. With large cache space, SR increases with the increase of request arrival rate. This
can be explained as follows. With a large cache space, a node can obtain more cache copies when the request arrival rate is high. Then, SR would be also high.

However, if the cache space is too small, under a high request arrival rate, the network traffic is busy and all messages suffer from the long delay and more replies cannot be delivered in time. That tells the reason why SR slightly reduces in Fig. 10-(c) and Fig. 10-(d).

Among the four algorithms, the non-anonymous algorithm is still the best and the simple anonymous algorithm is still the worst. This can be explained by the results of PL and AD.

The anonymous algorithm with plain data and the full anonymous algorithm achieve quite similar SR. Although the later is better than the former in most cases, but the difference is not so obvious as in other metrics. The former even achieves a little higher SR in a few cases.

VI. CONCLUSION AND FUTURE WORK

This paper studies how to achieve anonymity in cooperative caching. We design an anonymous caching algorithm, which borrows idea from the onion routing protocol. Data query and access are realized based on anonymous hop-by-hop broadcasting. Pseudonyms are used in communications so as to protect the real id. We analyze the security level of the proposed algorithm and both sender anonymity and receiver anonymity are achieved. We also conduct simulations to evaluate effect of anonymity mechanism on performance. Different variants of anonymous caching algorithm and non-anonymous algorithms are simulated for comparisons. The results show that anonymity will cause additional message cost and time cost, but the cost is reasonable.

To the best of our knowledge, this is the first work on anonymity in cooperative caching, further study is obviously necessary. We will improve our work by considering design with identity-based encryption (IBE). Real implementation with field testing is also interesting and significant work.

ACKNOWLEDGMENT

This research is partially supported by The National Natural Science Foundation of China (No. 61202396), Guangdong Natural Science Foundation (No. S2012010010870), Pearl River Nova Program of Guangzhou (No. 2011J2200088).

REFERENCES