# Tightly Cooperative Caching Approach in Mobile Ad Hoc Network

NWE NWE HTAY WIN[1,3], BAO JIANMIN[2], CUI GANG[1], DALAIJARGAL PUREVSUREN[1]
School of Computer Science and Technology[1], College of Internet of Things[2]
Harbin Institute of Technology[1], Nanjing University of Posts and Telecommunication[2], University of Computer Studies (Mandalay) [3]
Harbin[1], Nanjing[2], Mandalay[3]
CHINA[1, 2], MYANMAR[3]
bao@njupt.edu.cn

*Abstract:* - Mobile Ad hoc Network (MANETs) relies on cooperation of all participating nodes. All mobile nodes may easily vulnerable to selfish/malicious nodes which can lower down data communication and degrade the network performance due to refusal of relaying packet in order to save their own resources such as power, time, etc. These non-cooperative behaviors become challenging in cooperative caching system which mainly depends on cooperative nodes. To solve this problem, we propose tightly cooperative caching approach (TCCA) to encourage all nodes to aggressively cooperate by using enhanced credit-based distribution algorithm based on the resource status of the mobile nodes and the demand volume of the request to have load balancing among the acceptor nodes and requester nodes. According to the experimental results, our approach gets the superior results than other approaches under different parameter setting of nodes and network structure.

*Key-Words:* - mobile ad hoc network, cooperative caching, credit-based distribution, load balancing

## 1 Introduction

The mobile ad hoc networks (MANETs) are impromptu wireless communication among mobile nodes. MANETs allow users to access and exchange information regardless of their geographic position or proximity to infrastructure. They offer an advantageous decentralized character to the network in the absence of a fixed infrastructure [1]. Decentralization makes the networks more flexible and more robust.

However, these advantageous factors have become the challenging issues because connection can rapidly change in time or even completely disappear. Nodes can appear, disappear and re-appear as the time goes on and all the time the network connections should work between the nodes that are part of it. So, the situation in ad hoc networks with respect to ensuring the data accessing and connectivity is aggressively demanding.

Most of the researches [2][3][4] on MANETs propose on the development of dynamic routing protocols for connectivity strength among mobile nodes. Although routing is an important issue in ad hoc networks, other issues such as data access are also very important since the ultimate goal of using such networks is to provide data access to mobile hosts [5]. Therefore, there are still several challenging issues in data accessing on MANETs due to frequent disconnections, high mobility of mobile nodes, limited bandwidth utilization and the poor power resources such as battery life and storage capacity.

To overcome these challenges on data access in wireless ad hoc networks, data caching is one of the most attractive techniques that can increase the efficiency of data access [6-8]. Data caching means that mobile nodes copy some parts of data they want from the data source and store it in its local memory. Due to caching or holding some pieces of data, the mobile nodes will not need to request to the server whenever it receives data requests. In this way, the total access delay is decreased because of the service provided by these cache nodes.

However, caching techniques used in one hop mobile environment may not be applicable to multi-hop mobile environments since data or request may need to go through multiple hops. Caching alone is not sufficient to guarantee high data accessibility and low communication latency in dynamic systems with limited network resources. As mobile clients in ad hoc networks may have similar tasks and share common interest, cooperative caching which allows the data and information to share and collaborate each other among multiple nodes, can be used to reduce bandwidth utilization, power resource consumption of mobile nodes and increase access latency. Furthermore, cooperative caching technique

allows mobile caching nodes to coordinate the caching data tasks such as the redirection of data requests, cache placement, cache replacement and cache invalidation process.

Although previous cooperative caching schemes [9][10][11][12] have addressed to accessibility of data objects on multiple hop client caches, they miss the important facts on cooperative caching is whether all mobile nodes tightly participate their cooperative works and load balancing over the nodes or not. In reality, it is impossible and impractical to assume that all mobile nodes on the network will follow this cooperative behavior in the cooperative network such as MANETs, which mainly rely on cooperative behavior of all participating nodes.

Nodes may be selfish, lazy or greedy that is they may refuse to relay packets for other nodes in order to save their own resources including energy, bandwidth, computing capacity or available memory in civilian applications. Those kind of nodes may refuse the packet delay for others if and only if they expect benefit of doing so is larger than that of acting cooperatively and vice versa [1].These non-cooperative behaviors can not only lower down the data passage on the network but also even break down the network.

To facilitate the non-cooperative communications among such kind of mobile nodes in ad hoc networks, many mechanisms on packet routing or forwarding has been proposed in [14][15][5]. But, there are a few works in caching approach considering selfish node behaviour proposed by the researchers [8] [16] using game-theoretical algorithms but they didn't address to cooperative caching system. In our proposed system, we skim this credit-based distribution mechanism [14][15] and apply it on cooperative caching system. Besides, in our system, we propose Tightly Cooperative Caching Approach (TCCA) to guarantee the balanced credit-distribution and load balancing between group nodes. So, we are the first to our knowledge to propose tightly cooperative caching mechanism applying credit-based distribution in penalty and reward incentive manner and load balancing features on various malicious nodes by analyzing the current resource status of acceptor nodes and the demand volume of the requester nodes.

The rest of the paper is organized as follows. Section 2 presents related works. The problem model and its definition are described in Section 3. The detailed description of TCCA caching approach is discussed in Section 4. Section 5 evaluates the system performance and the paper concludes in Section 6.

## 2 Related Work

A lot of researches have been designing the caching mechanisms in mobile ad hoc environments. But, only few researchers have been paying attention on cooperative caching to impose more benefits of mobile ambient. Among them, the area of selfish removal caching system in cooperative caching has not received much attention. The researchers in [17] proposed CoCa, a cooperative caching protocol to share mobile cache contents to reduce both the number of server requests and the number of access misses. They proposed GroCoCa in [18] which extends CoCa in the cost of extra power consumption. The researchers in [5] designed and evaluated three caching techniques, viz., Cache Data which caches the passing-by data item, CachePath which caches the path to the nearest cache of the passing by data item, and HybridCache which caches the data item of its size is small enough, else caches the path to the data. Zone Cooperative (ZC) is proposed in [7] which search the data in the zone before forwarding the request to the next client that lies on the path towards server. However, the latency may become longer if the neighbours of intermediate nodes do not have a copy of the requested data object for the request. In our system, it can reduce latency because we can know which node hold what data item by using the list of group table.

An effective cooperative cache replacement policy is proposed by the research work [11] for mobile P2P environments based on the sizes of the data objects. They proposed to place smaller objects in the local cache of each peer and larger objects in idle neighbours peer. This scheme will cause increase in caching overhead in the case of a busy cache must always request their large data needs to the idle peer. The researchers in [9] proposed a cooperative caching scheme called COOP for MANETs addressing two problems, cache resolution and cache management. The former problem used cocktail approach which consists of two basic schemes: hop by hop resolution and zone-based resolution. In cache management, COOP uses the inter-category and intra-category rules to minimize duplicate data in caching. But, this scheme can flood in introducing extra discovery overhead.

The work in [19] proposed proactive approach in that they will cache the data of leaving nodes depending on the caching information table (CIT). Zone manager of each group will decide which data

is to be cached. Our approach also proposes this feature to cache the data of leaving nodes by extending in the fact that a node which has rich resources among the group nodes will accept the cache data of leaving nodes. All the nodes in the group know which node are the rich resources among them by using cache data exchange message. So, we can eliminate the role of zone manager.

Distributed selfish replication and caching of multiple objects is studied in [20]. In their model, the set of objects are not in the network initially and the caching cost is not considered. The study in [16] presented caching system on selfish MANETs in paying for cost to the service provider for caching service adopting game theory. But, they addressed in simply caching not for grouping. The next data caching considering selfish MANETs [8] also presented their approach based on game theoretical analysis. They considered distance-dependent caching cost for selfish MANETs but they are topology dependent. Some researchers [20][16] proposed caching in selfish MANETs but they didn't propose how cooperative caching system is implemented to be tight. In our system, we present caching system in cooperative behavior to overcome the problems of selfish MANETs in load balancing among selfish, lazy and greedy nodes and in cooperation with tightly manner among those nodes.

# 3 Problem Model and Definition

In this section, we describe the problem model and its definition applied by our mechanism, viz., network model, mobile computing model and grouping model.

## 3.1 Network Model

The network topology of ad hoc network can be represented by an undirected connected graph G = (V, E), where V is the set of mobile nodes in the network connected with edge or link E. The existence of a link (u, v) $\in$ E also means (v,u) $\in$ E, and two network nodes communicate directly with each other, which is represented by an edge on the graph, whereas nodes which cannot communicate with each other may still contend directly with each other, due the shared nature of wireless medium. If mobile nodes u and v are within the transmission range of each other, they can be thoughts as one-hop neighbors. The combination of mobile nodes and transitive closure and connectivity of their one-hop neighbors forms a mobile ad hoc network.

## 3.2 Mobile Computing Model

In a mobile computing system, the geographical area is divided into small regions, called cells. Each cell has a base station (BS) and a number of mobile hosts (MHs). BSs manage the communication between inter-cell and intra-cell. The MHs communicate with the BS by wireless links. An MH can move within a cell or between cells while retaining its network connection. An MH can either connect to a BS through a wireless communication channel or disconnect from the BS by operating in the doze or power save mode. The mobile nodes of our caching mechanism are modeled as shown in Fig 1.

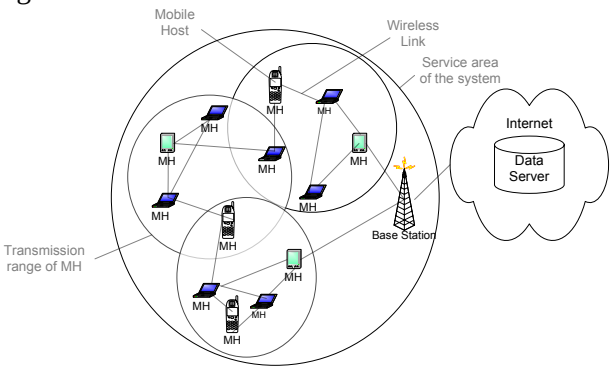


Fig.1 System Structure

## 3.2 Grouping Model

A hop represents one partition of the path between source and destination. Each MH and its one-hop neighbors can be formed as a group. Each MH in a group is in the range of coverage or transmission area of other MHs in that group. Each MH has a group member ID which may be IP address or unique host IP. For the connectivity between the nodes, we use a periodic beacon of "Hello" message, known as "Keep-Alive-Signal". By doing so, each MH can know who its one-hop neighbor are. We can extend the k-hop neighbor by piggybacking the (k-1)-hop neighbor information in "Hello" messages. When the cache miss occurs from its local cache or group cache, the data request can be sent to other 1-hop neighbor along to the path of data source. So, each MH only needs to maintain information of one-hop neighbor.

# 4 Problem Solving

This section describes how cooperative caching mechanism is implemented to be tight on various kinds of nodes nature and how the nodes on the network are managed to balance in caching

workload by using credit value distribution algorithm.

### 4.1 Network Model

One of the most important issues in designing caching system on MANETs is how to deal with selfish or malicious nodes called lazy and greedy nodes which disturb the data flow of the network and how to encourage them to take part in caching cooperation by giving payoff such as reward or penalty. Depending on the common nature of nodes, we classify the mobile nodes in three groups: lazy, greedy and selfish node as following description.

- Lazy Nodes : Nodes that do not have any work in current time and also do not desire to help other nodes' request and even they do not have any plan to request other nodes. According to the system, if a node holds enough credit value and also its node have no activation in any work, we will punish those kinds of nodes to join cooperation by reducing its credit values to the starting initialization value.
- Greedy Nodes : Nodes that have intense work, excessive desire in acquiring information, data request by saving their rich resources. So, they are eager to ask some nodes for their every needs. It can cause overburden for other nodes with their jobs and this will be unbalanced load balancing over the nodes because of its glutton. Our TCCA can overcome this problem by levying some credit value for asking request to others. Every node will decrease credit-value depending on its demand for asking helps. Therefore, according to our algorithm, they do not dare to ask so many requests to other without actually need because they realize effectiveness of their credit-values.
- Selfish Nodes : Nodes that do not forward other's packets and data requests so, they can maximize their benefits at the expense of all others. They want to preserve their resources and energy rationally. We will punish those kinds of nodes due to non-cooperation according to the rules of our approach described in Section 4.3.

### 4.2 Caching Tables

There are two types of table which need to be kept by every mobile node in our mechanism. One is to record its own current information and called as Self-Table {node_id, cached_data_id, cached_data_item, timestamp, credit_value, resource_value} and the another one is MemberTable {member_node_id, cache_data_id, timestamp, member_credit_value} in order to keep the caching status of group caches. The timestamp of both tables is used to know the caching time of cache data for checking cache validation. The credit_value and the member_credit_value are used for credit value distribution mechanism and the resource_value is used to check the workload status for load balancing purpose. Each node knows the updated cache data information of each other by periodically sending the cache data exchange message and update their local and group cache tables according to the group cache information on this exchange message. As an advantage of keeping these tables, when a node requires a data request, it just needs to send the one which is caching its requested data such as group cache or data source. Moreover, due to the knowledge of cache data item among group members, we use cache admission control by caching distinct data among group members. Therefore, it can save the data redundancy among group caches. So, it is obvious that our mechanism significantly improves the caching performance by decreasing the caching overhead and cache query delay. Moreover, by using credit-value retained in each node, whenever the mobile nodes encounter the data requests of other nodes, they order the data requests according to the credit-value owned by the requester nodes and offer the request first to the one which holds the highest credit value. So, the higher the credit value owned by the nodes, the more chances they can get the offer. This process forces the nodes to participate the caching operation in order to get the higher credit values.

### 4.3 Tightly Cooperative Caching Algorithm (TCCA)

This section describes the cooperative caching mechanism which enforces all the nodes to take part in caching by using credit-based distribution approach and be able to adjust the workload among the nodes by using load balancing control.

The major work of credit-based distribution approach is to enforce the nodes to cooperate the caching process by giving incentive for their cooperation and non-cooperation. However, ordinary credit-based distribution algorithm can lead to unbalanced credit distribution because there are various kinds of resource limited devices on mobile networks and the capabilities they can performed will be different depending on the requests they

received. Moreover, due to the resources condition they owned (poor or rich resource) and the demand type they received (light or heavy demand), the works related with caching abilities such as storage, relaying packets, transmission message, etc. may be overburden or idle for some nodes which leads to unbalancing the workload. In order to control this load balancing capabilities on credit-based distribution approach, we enhance the credit-based distribution approach by taking into account of resources of the acceptor nodes and the demand type of the request in distribution the credit value for increasing the credit value for cooperation and decreasing the credit for non-cooperation.

#### 4.3.1 Resource and Demand Comparison

To impose the balanced credit distribution on the mobile nodes, we analyze the resources of the acceptor nodes and the demand type of the requester nodes depending on their related criteria.

The resource value denoted as $C_r$, is calculated on the resource related criteria such as available battery life ($Bl$), memory space ($Ms$) and processing power ($Pp$) of a node $N_i$ according to Eq (1). All of these criteria are considered at time $T_c$ when a request is received.

$$C_r(N_i) = \alpha * Bl_i + \beta * Ms_i + \gamma * Pp_i \quad (1)$$

Wherein α, β and γ are the tunable parameters to indicate the weight of three criteria. Setting the proper value to the weighting factors achieve the better performance in a design issue.

The demand value $C_d$ is evaluated on the criteria which is related to the data demand of the requester node such as size of demand ($Sd$), access probability of demand ($Ap$) and data retrieval delay ($Rd$) of that demand over the network. Although the size of the demand can easily be known, the other criteria need to be derived from other facts. The access probability is similar with the popularity of the cache item [21]. It means that the most popular data item can easily be fetched from any node because most cache hosts have willingness to cache the data that is frequently requested by most of the mobile nodes. For the data retrieval delay criteria, it will vary depending on the several factors such as node location, packet size, the number of hops to reach a node and network congestion [22]. However, the evaluation based on too many factors cannot improve the better result due to the difficulties of deriving values. So, we account only number of hops to calculate the average query delay and the specific time taken to reach a hop.

To get the expected number of hops between any two nodes (the node that needs data and the node holding that data), we use stochastic geometry applied, the probability density function of s [21] is applied as shown in Eq (2)

$$f(s) = \frac{4s}{a^2 b^2}\left(\frac{\pi}{2}ab - as - bs - 0.5s^2\right) \{0 \le s < b < a\} \quad (2)$$

We assume this probability function is in a rectangular topology with area a × b and uniform distribution of nodes in the transmission range, $r_0$. The E[H], the expected minimum number of hops between any two nodes in the network is equivalent to dividing E[S], the expected distance, by $r_0$. The E[H] needs to be assumed a lower bound because the nodes are sparse on the network and the number of hops will inevitably increase because of routing through the longer distance to reach a certain node. When $a = b$, the expected number of hops is derived from Eq (3) [2] as shown below.

$$E[H] = 0.521 * a / r_0 \quad (3)$$

We derive data retrieval delay by using Eq (4), the expected number of nodes between any nodes by multiplying the specific time, $T_h$, delay time to reach a hop. So, we derive data retrieval delay $r_i$ of a node $N_i$ as shown in Eq (4).

$$r_i = E[H] \times T_h \quad (4)$$

Depending on the derived values of criteria for demand value, we calculate the demand value of a node $N_i$ according to Eq (5) as follows.

$$C_d(N_i) = \alpha * Sd_i + \beta * Ap_i + \gamma * Rd_i \quad (5)$$

Depending on the evaluation values: resource value $C_r$ and demand value $C_d$, we calculate the comparison value ($V_j$) according to Eq (6).

$$V_j = C_d / C_r \quad (6)$$

In the case of measurement in $C_d$ and $C_r$, we assume that they are similar unit to analyze their value. The core work of this comparison value is to judge the credit value distribution on the mobile nodes to increase or decrease the credit value from the current credit value currently hold by the mobile nodes. The detailed process of enhanced credit-based distribution is described in next section.

#### 4.3.2 Credit-based Distribution

Our credit-based distribution algorithm on cooperative caching mechanisms can bring many benefits to ad hoc network by preventing overloading caching work on some weak nodes, by forcing the strong nodes to take part in caching process due to the penalty imposed and by giving fair payoff on different kinds of nodes in different situation which connect on the volatile network structure such as strength or weak network bandwidth and so on.

Starting from the caching mechanism, we assign the default credit values to every node. Generally, if a node cooperates in caching process, it will receive certain credits ($V_j$) and plus that payoff to its own credit value ($V_c$) as a reward. If a node chooses not to cooperate, the payoff is the penalty by decreasing its current credit value by subtracting the certain credits. However, the common credit distribution is obviously unfair on diverse resource condition of mobile nodes. In order to impose balanced credit distribution, we take into account of the two types of the nodes, poor node and rich node depending on the enrichment of resources they owned, in other ways, the higher or lower value of $C_r$ and two types of demands, light demand and heavy demand depending on the volume of requested data which can consume the energy or memory for the acceptor nodes.

Depending on the nodes (poor or rich node) which are facing with different demands (heavy or light demand), the payoff such as penalty and reward should be different because it will not be unfair to punish the same payoff when a poor node refuses the heavy demand and a rich node denies a light demand. Like this way, it should be different payoff in rewarding when a poor node relays the heavy demand and a rich node helps in relaying the light demand. For same level between resource and demand of mobile nodes, the payoff will be common as general credit-based distribution approach.

Our credit value distribution function for cooperation and non-cooperation chosen by the nodes are defined as described in Eq (7) and depending on their choice, the current credit value ($V_c$) is increased or decreased by the comparison value $V_j$.

$$V_c = V_c \pm \mu V_j \quad (7)$$

Wherein $\mu$ is the tunable variable to balance the credit value depending on the comparison value $V_j$ between the poor or rich resources and light or heavy demands. When the same level of resource and demand, in other ways, the value of $V_j$ is equal to 1, the value of $\mu$ is also 1 by using normal credit distribution on the common resource and demand nature. For the unbalanced resource and demand condition, the value of $V_j$ will be lower or higher than 1, in this case, the parameter configuration of $\mu$ changes depending on the value of $V_j$ and payoff condition, penalty or reward. In our credit distribution, we select the value of $\mu$ as 0.2 for penalty and 0.8 for reward between poor node and heavy demand, and choose the value of $\mu$ as 0.8 for penalty and 0.2 for reward between rich node and light demand. Most of scholars evaluate the weight value using intellectual algorithm via iteration. We determine our weight value $\mu$ according to the empirical value based on resource and demand status of all nodes in the network. On the mobile nodes which choose cooperation or non-cooperation, detailed calculation of our balanced credit-based distribution are described in Table 1 by assuming the current credit value $V_c$ of all nodes is 6.

**Table 1**. Computation of Payoff for Penalty and Reward depending on Resource and Demand

| Comparison Value $V_j = C_d/C_r$ | Parameter of $\mu$ | | Resource and Demand Pair | After PayOff | |
|---|---|---|---|---|---|
| | For Penalty | For Reward | | After Penalty | After Reward |
| Vj=1 (3,3) | μ=0.5 | μ=0.5 | (poor, light) & (rich, heavy) | $V_c$ =5.5 | $V_c$ =6.5 |
| $V_j$ >1 (5,3) | μ=0.2 | μ=0.8 | (poor, heavy) | $V_c$ =5.7 | $V_c$ =7.3 |
| $V_j$ <1 (3,5) | μ=0.8 | μ=0.2 | (rich, light) | $V_c$ =5.6 | $V_c$ =6.1 |

### 4.4 Cache Placement and Replacement Control

Storage management on data object in the cache is one of the important functions of data caching because of limited cache size and impact on system performance. So, it needs to cache the most useful and essential data which is often requested by the nodes. When the cache has the free space for new

data, we just place the new data on the cache. If the cache is full of cache data, one object has to be removed from the cache to make room for the data that has to be brought in. So, we need to consider which object is removed for new data. System performance will be better if we choose an object that is not heavily used.

Many cache replacement algorithms have been developing by the researchers with their corresponding parameters for their objectives. However, if we are using many parameters for finding the value function, it is not easy to focus the better performance. Therefore, in our replacement control in caching database, we use the most common usage criteria in the caching system such as access frequency denoted as $a$, data size denoted as $s$ and recency denoted as $r$ of the cache data item $C_i$.

The reason of taking into account of access frequency is that, the more frequent requested data item can be assumed that it will be requested in the near future. Therefore, we want to evict the less frequent data item to replace the data item that is high demand or popularity among the mobile nodes.

To get more cache hit, a data item with larger data size should be chosen for replacement. By choosing larger data which has taken the more cache space, the cache can accommodate the more data items and satisfy more access requests.

Recency is based on the observation that data that have been heavily used recently will probably be used in the near future. Conversely, the data that have not been used for ages will probably remain unused for a long time. So, our system finds minimum recency value of data item to remove from the cache. Based on the above function, we find the value of cache data item $C_i$ using the following Eq (8).

$$value(C_i) = w_1 * a_i + w_2 / s_i + w_3 * r_i \qquad (8)$$

wherein $w_1$, $w_2$ and $w_3$ are the weighting value. We find the replacement value of all cache data item $C_i$ of a node and we evict the data item with the lowest value to make a free space for new data item to replace in the cache.

### 4.5 Complexity Analysis of TCCA Approach

In this section, we analyze time complexity for searching cache data item on the network. According to TCCA algorithm, there are four places to search cache data for a node when it needs to make a data request. They are local cache, a place which almost needed data of a node is stored, group cache, the places which stores most of all needed data of the group neighbours, remote cache, the places which keeps the data often demanded by one-hop neighbour and global cache, a place which stores all requested data on the network, known as data source.

To analyze the complexity of data discovery process, the searching time complexity of local cache is $O(1)$. For group cache search, a node $N_i$ in a group $G_k (1 \le G_k \le M)$, it needs to send only one node that holds the wanted data, so it will take $O(1)$. Instead, the other system, node $N_i$ sends all other group nodes $N_j = \{j : 1 \le j \le N-1\}$ such that $i \ne j$. So, it is obvious that their system will take $O(N)$. For remote cache searching, our system will take $O(1) \times M$ so $O(M)$ in remote cache time complexity. Apparently in this case, the other approaches take $O(N^M)$ for remote cache. For global cache discovery process, in other ways, data searching on the data source, we take $O(1)$ for all types of the system.

Additionally, data request to other nodes, other groups and data sources, will generally take $O(M)$ time complexity in our system but in others, they will take $O(N + N^M)$. The time complexity will be the same for two ways in requesting and receiving the data. So, our system will eliminate traffic congestion on the data passage between the nodes and also reduce query retrieval delay and bandwidth consumption for searching cache data item on the network.

## 5 Problem Evaluation

We demonstrate our caching mechanism using NS2 simulator with DSDV routing protocol for routing services. A set of mobile nodes are moved within a rectangular workplace in an area of 1000x1000 $m^2$ for 100 seconds simulation time based on reference point group mobility model. We assume that data server will keep all the data items which will be requested by the mobile nodes with 1500 bytes of data item size [23]. To cache the fresh data, we use TTL-based cache consistency strategy with 5s timeout value. Every mobile node acts as a client node else one node is left for acting as data source. Different numbers of mobile nodes are used for each experiment with different parameter setting. Each client node sends their read-only queries to the data

source connected on randomly generated network topologies.

We evaluate the performance of our TCCA caching mechanism compared with cooperative caching schemes: CoCa (Cooperative Caching) [24], COACS (Cooperative and Adaptive Caching Systems) [22] and with non-cooperative caching scheme called N-CoCa (Non-Cooperative Caching) which is a conventional caching scheme without applying any cooperation mechanisms. These caching mechanisms are analyzed under the various essential performance metrics under following different impact of parameters.

## 5.1 Impact of Group Size

This section explores the influence on cache hit ratio and average query delay under various group sizes (1, 5, 10, 15, 20, 25, 30). The number of nodes for each group is equally assigned depending on the node density of the network. The cache hit ratio means the percentage of accesses that results in cache hit from its local cache or its group caches. The average query delay means the time interval between the time of generating the data request and the time of receiving the data offer from any source such as local cache, group cache or data source.

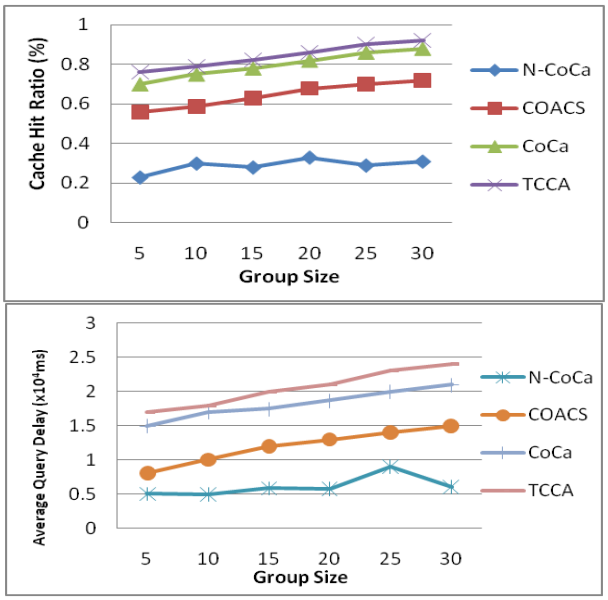


Fig 2. Impact of group size

As shown in Fig 2, all the cooperative caching schemes improve their cache hit ratio and average query delay because the higher the node density within a group is, the more neighbor nodes in a zone can share their local cache data to each other. As a result, the total cache hit ratio is improved and the average query delay is reduced due to the fact that a global cache hit incurs shorter query time from the data source. Our TCCA outperforms the other mechanisms because of well management on the nodes to be tight on caching cooperation. For the performance of CoCa and COACS, the CoCa gets better performance than COACS and N-CoCa because COACS is lack of improving on cache replacement or other caching mechanism else cache discovery process and N-CoCa could not enlist the cooperative caching.

## 5.2 Impact of Node Nature

In this section, we investigate the data distribution and workload which can be performed by the mobile nodes under different misbehavior of nodes nature. The objective of this experiment is to show how the nodes are implemented to tightly cooperate each other by analyzing the data distribution pattern which passes the demand and offer among neighbor nodes and workload percentage they can perform on their own work and other neighbor nodes' work. In this case, the workload means the amount of work they can perform, it does not relate with load balancing which is the amount of work assigned on each node.

In order to show the performance of TCCA compared with other caching mechanism on the impact of nature of nodes, we simulate the mobile nodes in the network to behave selfishly and maliciously by periodically changing their custom properties under different situations such as workload, network topology, and network delay and so on.

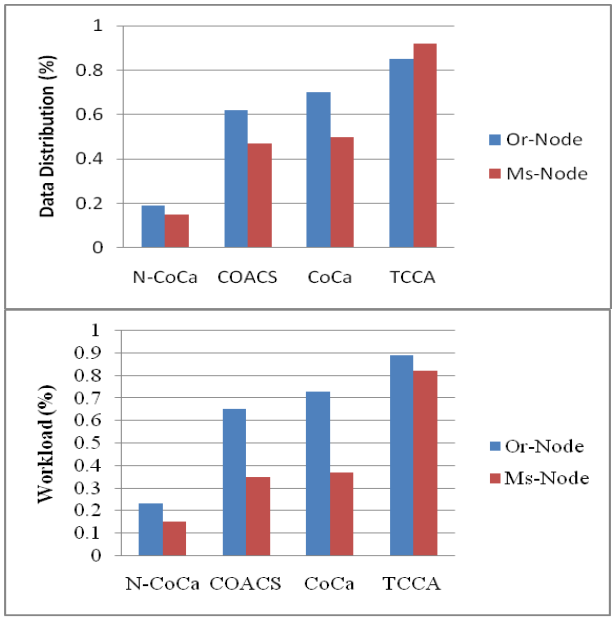


Fig 3. Impact of Nature of Nodes

Depending on the nature of nodes: ordinary node and malicious node; we can see from the Fig 3. that in both experiments, TCCA outperforms other by getting significant highest results on both ordinary and malicious nodes due to the effectiveness of tightly cooperative caching algorithm. The other cooperative caching mechanisms COACS and CoCa performs well on ordinary nodes like TCCA but they cannot take any advantage from their system on malicious nodes due to their lack of consideration on nature of nodes. Apparently, N-CoCa performs the worst among the caching mechanisms for every kind of nodes because of its simple working on caching mechanism.

Compared with two charts of Fig 3 on ordinary node (Or-Node) and malicious node (Ms-Node), the malicious nodes can distribute the data more than ordinary node because of cooperative policy: reward and penalty but they cannot perform more workload than ordinary node because of their own heavy workload. So, obviously, although the malicious nodes manage well the data distribution process, they still need to keep pace in workload they can finish.

### 5.1 Impact of Cache Size

This section measures the two performance metrics: cache hit ratio and caching overheads under different cache sizes (50, 100, 150, 200, 250). Caching overheads means all the data passages in caching systems, viz., data demand, data offer and other message notification among the nodes for caching process.

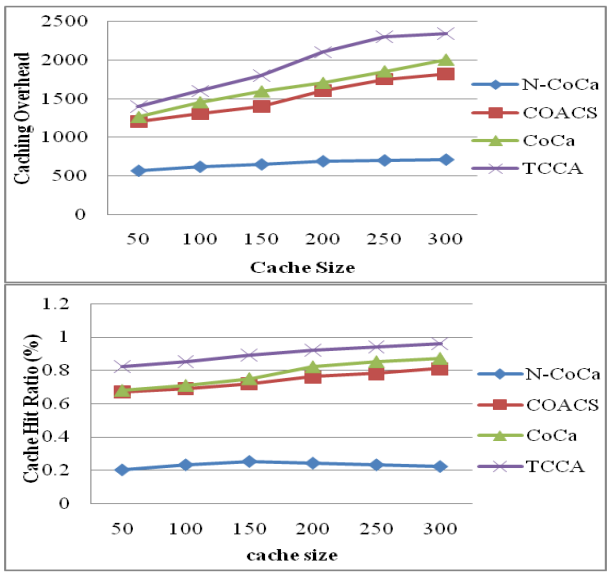


Fig 4. Impact of Cache Size

All mechanisms get better performance with increasing cache size in increase of cache hit ratio and decrease of caching overheads because the mobile nodes experience a higher local cache hit as the cache size gets larger and reduce the messages passing among them.

As shown in the Fig 4, our TCCA gets significant better results than other caching system in performance metrics because we reduce the chance of caching the same item in the local cache, group cache to reduce the redundant data items. Therefore, we got superior results in increasing cache hit ratio and decreasing caching overheads among data sources and client mobile nodes by applying tightly cooperative caching scheme. The other two cooperative caching schemes also get the better result in this measurement while the cache size is increasing but they are still cannot reach the same level like TCCA could due to the lack of considering on cache admission, placement control on cache data. The performance of N-CoCa is still modest on both metric types.

## 5 Conclusion

Cooperative caching is the effective scheme to support the data access on MANETs by reducing the network traffic, overlays and increasing data access rate. Most scholar works perform cache management mechanism with different perspective in various parameters and methods, but little attention is given to the issues of tightly encouragement on the mobile nodes. In this paper, we propose tightly cooperative caching mechanism by enhancing the credit-based distribution algorithm on the resources owned by the mobile nodes and the demands types of the data request and develop the cache placement and replacement control mechanisms based on critical criteria evaluated on the cache data item. As the advantages of our proposed mechanism, firstly, it can encourage all the nodes to participate the caching tasks and then it also adjusts credit-values based on the resources and demands to have balanced credit distribution. Secondly, due to the priority on the credit-values, it has already solved the problem in contending for data accessing among requested nodes. Thirdly, if a group member node leaves from a group, the efficiency of grouping caching cannot be effected because the cache data of leaving nodes are transferred to the one of member nodes which posses rich resources among them. Furthermore, this algorithm uses cache admission control to reduce redundancy for duplicated caching data within the

same group such that the cache space can be used to accommodate more distinct data items. Lastly, it makes all the nodes to have balanced workload and reduce overburdening on a node for other demands. In conclusion, we analyze the performance of our proposed system under different impact of experiments over unpredictable nodes nature of dynamic network. Experimental shows that our proposed TCCA algorithm can significantly improve the overall performance of caching approach when compared to other caching approaches.

## Acknowledgements

This work is supported by National 973 Program of China (2011CB302903); NSFC (61100213); Specialized Fund for the Doctoral Program of Higher Education (20113223120007); Program of Natural Science for Universities of Jiangsu Province(12KJD510007), Ministry of Education Key Laboratory of Development Fund projects under Grant No.NYKL201105 as well as Scientific Research Foundation of NJUPT under Grant No.NY209017, China.

*References:*

[1] H. Y. Hu and H. Hu, Optimizing Energy Consumption of Data Flow in Mobile Ad Hoc Wireless Networks, WSEAS Transactions on Computers, 7 (7), 2008.

[2] M. Frodigh, , P. Johansson and P. Larsson, Wireless Ad Hoc Networking – The Art of Networking without a Network, Ericsson Review, 2000, No. 4

[3] D.B. Johnson and D.A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, The Kluwer International Series in Engineering and Computer Science, Vol. 353,1996, pp. 153-181

[4] C.E. Perkins, E.M. Royer and S.R. Das, Performance Comparison of Two on-demand Routing Protocols for Ad Hoc Networks, IEEE Infocom, 2000, pp. 3-12

[5] Y. L. Zhong and C.G. Gong, Supporting Cooperative Caching in Ad Hoc Networks, IEEE Infocom, 2004, pp. 2537-2547

[6] N. Chand, R.C. Joshi and M. Misra, Efficient Cooperative Caching in Ad Hoc Networks, IEEE COMSWA, 2006, pp. 1-8

[7] N. Chand, R.C. Joshi and M. Misra, Cooperative Caching in Mobile Ad Hoc Networks Based on Data Utility, International Journal of Mobile Information Systems Vol. 3, No. 1, 2007, pp.19-37

[8] Y. Chen and B. Tang, Data Caching in Ad Hoc Networks Using Game Theoretic Analysis, Proc. SUTC, 2010, pp. 43-49

[9] Y. Du and S.K.S. Gupta, COOP- A Cooperative Caching Service in MANETs, Proceedings of the IEEE ICAS/ICNS, 2005, pp. 58-63

[10] K. Shanmugavaidvu and M. Madheswaran, Caching Technique for Improving Data Retrieval Performance in Mobile Ad Hoc Networks, International Journal of Computer Science and Information Technologies, Vol .1 , No. 4, 2010, pp. 249-255

[11] J.W. Song, K.S. Par and S.B. Yang , An Effective Cooperative Cache Replacement Policy for Mobile P2P Environments, ICHIT IEEE, Vol. 2, 2006, pp. 24-30

[12] Y.W. Ting and Y.K. Chang, A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: Group Caching, IEEE Conference on Networking, Architecture and Storage, 2007, pp. 62-68

[13] D. Qian, C. Zhou and J. Zhang, Cooperation Enforcement in Ad Hoc Networks with Penalty, IEEE Conference on Mobile Adhoc and Sensor Systems, 2005, pp.179

[14] L. Buttyan and J.P. Hubaux, Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks, Journal of Mobile Networks and Application, Vol. 8, No. 5, 2001, pp. 579-592

[15] V. Srinivasan, P. Nuggehalli, C.F. Chiasserini and R.R. Rao, Cooperation in wireless ad hoc networks, Proceedings of Infocom , Vol. 2, 2003, pp. 808-817

[16] J. Zhai, Q. Li and X. Li, Data Caching in Selfish MANETs, Proc. ICCNMC, Vol. 3619, 2005, pp.208-217

[17] C.Y. Chow, H.V. Leong and A. Chan, Peer-to-Peer Cooperative Caching in Mobile Environments, 24th Int'l Conf. Distributed Computing Workshops (ICDCSW '04), 2004, pp. 528-533

[18] C.Y. Chow, H.V. Leong and A. Chan, Group Based Cooperative Cache Management for Mobile Clients in Mobile Environments, Proc 33rd Int'l Conf. Parallel Processing (ICPP' 04), Vol.1, 2004, pp. 83-90

[19] K. Prashant, C. Naveen, A. Lalit and C. Narottam, Proactive Approach for Cooperative Caching in Mobile Adhoc Networks, IJCSI, 2010

[20] N. Laoutaris, O. Telelis, Zissimopoulos, V. and I. Stavrakakis, Distributed Selfish Replication", IEEE Transactions on Parallel and Distributed

Systems', Vol. 17, 2005, No. 12, pp. 1401-1413

[21] C. Bettstetter and J. Eberspacher, Hop Distances in Homogenous Ad Hoc Networks, Proc. 57th IEEE Vehicular Technology Conf. (VTC-Spring 03), Vol.4, 2003, pp. 2286-2290

[22] H. Artail, H. Safa, K. Mershad and Z. Abou-Atme, COACS: A Cooperative and Adaptive Caching System for MANETs, IEEE Transcations on Mobile Computing, Vol. 7, No.8, 2008, pp. 961-977

[23] E. Chan, W. Z. Li and D. Chen, Energy saving strategies for cooperative cache replacement in mobile ad hoc networks, Pervasive and Mobile Computing, Vol. 5, 2009, pp. 77-92

[24] C.Y. Chow, H.V. Leong and A. Chan, Cache Signatures for Peer-to-Peer Cooperative Caching in Mobile Environments, Proc. 18th Int'l Conf. Advanced Information Networking and Applications (AINA '04), Vol. 1, 2004, pp. 96-101

[25] S. Zhong, J. Chen and Y.R. Yang, Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc-networks, IEEE INFOCOM, 2003, San Francisco, CA, Vol. 3, pp. 1987-1997