

Data Caching and Query Processing in MANETs

JINBAO LI

*School of Computer Science & Technology, Harbin Institute of Technology,
Harbin, China, 150001
Email: lijzh@hit.edu.cn*

YINGSHU LI, MY T. THAI

*Department of Computer Science & Engineering, University of Minnesota,
Minneapolis, MN 55455, USA
Email: {yili, mythai}@cs.umn.edu*

JIANZHONG LI

*School of Computer Science & Technology, Harbin Institute of Technology,
Harbin, China, 150001
Email: lijnbao@hit.edu.cn*

Received: July 15 2005; revised: August 20 2005

Abstract— This paper investigates query processing in MANETs. Cache techniques and multi-join database operations are studied. For data caching, a group-caching strategy is proposed. Using the cache and the index of the cached data, queries can be processed at a single node or within the group containing this single node. For multi-join, a cost evaluation model and a query plan generation algorithm are presented. Query cost is evaluated based on the parameters including the size of the transmitted data, the transmission distance and the query cost at each single node. According to the evaluations, the nodes on which the query should be executed and the join order are determined. Theoretical analysis and experiment results show that the proposed group-caching based query processing and the cost based join strategy are efficient in MANETs. It is suitable for the mobility, the disconnection and the multi-hop features of MANETs. The communication cost between nodes is reduced and the efficiency of the query is improved greatly.

Index Terms— MANET, mobile database, group-caching, join, query Plan

I. INTRODUCTION

The emergence of powerful portable computers, along with advances in wireless communication technologies, has made mobile computing a reality. Mobile applications such as stocks trading, traffic controls and weather forecasts have become increasingly popular. As the number of mobile applications increases rapidly, there has been a growing demand for the use of distributed database architectures for various applications. Various wireless data networking technologies, Wireless Application Protocols (WAPs) and the third generation mobile phones, have been developed recently. A mobile computer is envisioned to be equipped with more powerful capabilities, including the storage of a small database and the capability of

data processing. In a mobile computing environment, characteristics such as availability, connectivity, low-bandwidth, data quality, usage cost and the battery power impose new constraints on traditional distributed database systems. Traditional distributed database techniques cannot efficiently support queries in mobile computing environment. Consequently, query processing in mobile database, which is conducted by some fixed hosts and several mobile hosts, has emerged as an issue of growing importance [1].

A mobile database can be recognized as a kind of distributed database that supports mobile computing. In general mobile wireless networks [1], [5], there are two sets of entities: Base Stations (BSs) and hosts. The hosts are either fixed or mobile (called MH). Fixed hosts communicate over the network with a fixed topology, while mobile hosts communicate with other hosts (mobile or fixed) via a wireless channel.

Recently, the above mobile wireless networks are attracting more and more attention. Location-aware query processing technique has been studied in [2]. The authors proposed a cost model considering location, CPU and memory utilization, power consumption and transmission. The main problem is that the location information is needed and the support from the BS is required. Location-aware continuous queries are studied in [6]. Sanjay introduced a query processing model by using the hierarchical concept and summary database [7]. In this model, there is a summary database stored at each BS or MH. If the data referred by the query is un-accessible, some approximate results can be obtained from the summary database.

With the development of mobile computing, another kind of wireless network, MANET, emerges. In a MANET, both the hosts and the BSs are mobile. Thus, query processing becomes much more complex than that in general wireless networks [4]. How to optimize mobile queries, cache and replicate

This work was supported by the National Natural Science Foundation of China under Grant No.60473075; the Natural Science Foundation of Heilongjiang Province of China under Grant No.ZJG03-05 and No.QC04C40.

data, manage transactions and routing are the key issues in MANETs. Till now, most researchers just focus on the routing problems. Only a few researches studied query processing from the viewpoint of distributed database. To improve data access, three data replication methods were presented in [8]. A system framework for query processing in MANETs was proposed in [3]. This system only supports simple queries, and does not take the query optimization into consideration.

In this paper, the technologies of data caching and query processing in mobile distributed database systems are mainly investigated. The major contributions are as follows: (1) A group-caching based query algorithm is proposed, the purpose of which is to execute a query at a single node or within its group. At first, the algorithm searches the local caching index and obtains some partial results. Then, it searches the group through group-caching index, and obtains all the matching results and the nodes where they are. At last, all the other results are obtained from the original nodes. (2) A cost model and an optimization technique for the join operation are proposed, which focus on reducing the transmission cost. The locations of the nodes and the MSS support are not necessary in the strategy. It evaluates a query cost based on the size of the transmitted data, the transmission distance and the query cost at each single node. According to the evaluations, the nodes on which a query should be executed and the join order are determined. This is an optimal query plan.

The paper is organized as follows. Section 2 introduces MANETs and mobile distributed databases. Node grouping and dynamic the group-caching strategy are presented separately in Section 3 and Section 4. Group-caching based query processing is introduced in Section 5. The cost model and join processing technique are illustrated in Section 6. Experimental results are shown in Section 7. The paper is concluded in Section 8.

II. MANETS AND MOBILE DISTRIBUTED DATABASES

A MANET is a special kind of wireless network consisting of a collection of mobile hosts. There is no fixed infrastructure in a MANET. The hosts are connected wirelessly. Each host has mobility, so the topology of the network changes continuously. There are no BSs in a MANET. MHs communicate either through a single-hop or multi-hop routing. Each MH is a router and has the ability of conducting search dynamically, locating and connection restoring. The MHs can also re-configure the topology of a MANET dynamically, do distributed processing, share data and route messages among MHs. The architecture of a MANET is shown in Fig. 1. The nodes in the figure are MHs. A dashed circle centered at each node represents the area where the transmission from this node can be successfully received.

The mobile database system in a MANET is a dynamic distributed database system, which is composed of some mobile MHs. Each MH has a local database system. The architecture of the mobile database system is shown in Fig. 2.

Each node in the mobile database system may be either a client in some applications or a server in other applications. Each node can propose a query as a client and can also process

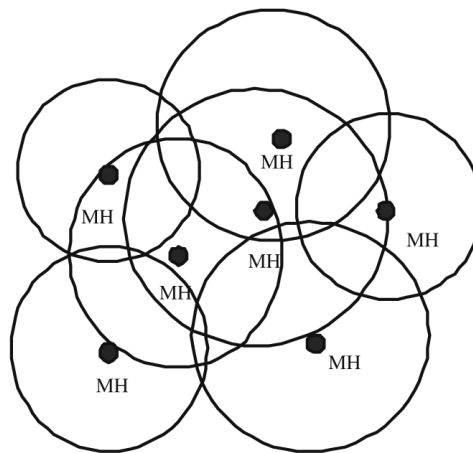


Fig. 1. MANET.

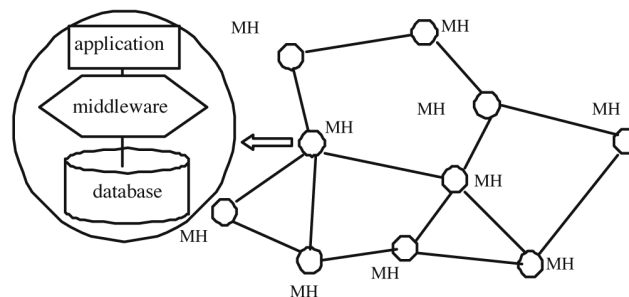


Fig. 2. Mobile Distributed Database in a MANET.

the queries from other nodes as a server. As shown in Fig. 2, each node needs a middleware to support the mobile query processing.

There are three layers in a mobile distributed database system: the application layer, the middleware layer and the database layer. The architecture is shown in Fig. 3. The application layer accepts user queries. The middleware layer is the core of the mobile distributed database system. Queries are processed by the middleware, transmitted to the middleware of other MHs in the network. The middleware of a MH sends queries to the local database system. After the database finishes executing a query, the results are transmitted from the middleware layer to the application layer and then the results are returned to the user. The middleware layer is transparent to users. Users need not to know the topology of the network, the status of the network and the query processing method. Users use the mobile database system as a local one.

The middleware layer is divided into three sub-layers further: the network layer, the cache layer and the query layer.

The network layer is responsible for the communications in the system. Its main functions are as follows. (1) Managing location information of nodes. Each node should clearly know its position and the relative position with the surrounding nodes. (2) Dividing nodes into groups. For ease of routing and caching, all the nodes in the system are divided into several groups according to their relative positions. When a node moves, the group information should be maintained. (3)

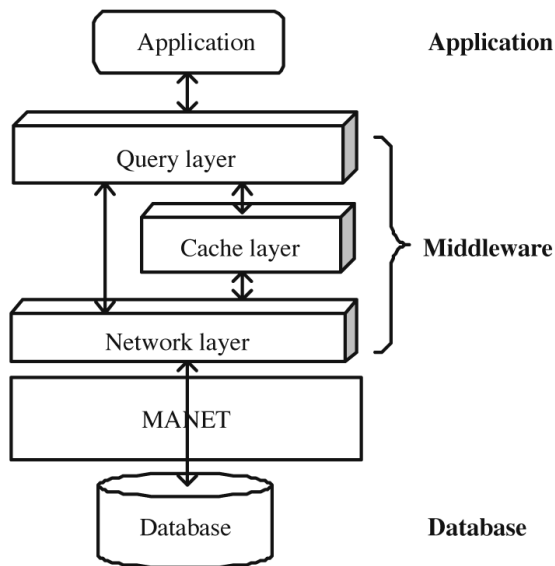


Fig. 3. Architecture of a Mobile Distributed Database System.

Routing data packets between the query layer and the cache layer. Group routing strategies are adopted in this paper to implement the communications between nodes that are not directly connected. The data from the query layer and the cache layer might be sent to any other nodes through the network layer.

The cache layer focuses on data caching in the system. The limited bandwidth in MANETs makes the communication cost the main cost of query processing. Node mobility and packets conflicts make the network topology change frequently, meanwhile, disconnection of nodes often occur. This delays the data transmission in the network. The cache layer stores data accessed frequently by the query nodes or their neighbors. This decreases the query response time and improves the data accessibility when node disconnection occurs. With the changing of network topology and the updating of the data at each node, the cached data should be updated dynamically to ensure the consistency between the cached data and the original data in the database.

The query layer parses the syntax of user queries and determines the query types. It searches a local or group cache index for the cached items. If all the query results are found, the query ends. Otherwise, the query layer searches other results from remote nodes. For join queries, the query layer evaluates the query cost and generates some distributed query plans. The data nodes and the query nodes select the optimized one based on the query cost and the status of the network.

III. NODE GROUPING

In a MANET, any node communicates peer-to-peer. In order to communicate with nonadjacent nodes, the communication path should be determined, which is called routing. Group routing strategy is adopted in this paper. It divides the nodes in the network into some groups. A routing request is sent to several different groups, and then the destination nodes are searched in these groups. This method broadcasts a request

among the groups, which reduces the broadcast range of the request packet.

Each group has a master node. It maintains the topology of the group dynamically. To reduce the cost for the master to maintain the group and increase the communication speed among nodes in the group, each master should communicate with its members directly through single-hop routing. A node needs not to know the whole topology of the network. But it needs to know its neighbors within a group. Each node determines which group it belong to through exchanging a series of messages and feedbacks with its neighbors. The rules for grouping are as follows:

- 1) Each group has only one master, and the group ID is the same as the master's ID.
- 2) Each group has some members. The distance between the master and its members is one hop.
- 3) Every node belongs to only one group. That is, each member has only one master.
- 4) The master is a member of itself.
- 5) The neighboring groups communicate with each other through their gateways. When member *i* of group A is a neighbor of some member of another group B, then node *i* is called the gateway of A→B. The Gateways are communication bridges between the neighboring groups. The information should be exchanged through the gateways among groups.

In general, any two nodes can communicate with each other through multi-hop routing in MANETs. According to the above rules, all the nodes in a MANET will be divided into several groups, and the members in a group can communicate with each other through the master of this group. These groups can connect with each other through gateways. Thus, any two nodes in a network can communicate with each other through masters and gateways. Because the distance between a master and its members is one hop, the distance between any two nodes in the same group is at most two hops. For two neighboring groups, the distance between two masters is not larger than three hops and the distance between any two nodes in these two groups is at most five hops.

If a group has just one node, the node must be the master and it is called an island node. If all the members in a group leave this group, the master in this group will become an island node. The node which is first added to the network is an island node.

The grouping steps are as follows:

- 1) Node A in the initial phase or after its position changing broadcasts a HELLO message (including node A's MasterID) to A's neighbors. The neighbors receiving the HELLO message answers a RHELLO message back to node A.
- 2) After node A receives the RHELLO message from its neighbors, it first updates its status:
 - a) If node A is a member of its group but not the master, then it determines by itself to leave this group or not. If A leaves the group, then it becomes an island node.
 - b) If node A is the master but loses all its members,

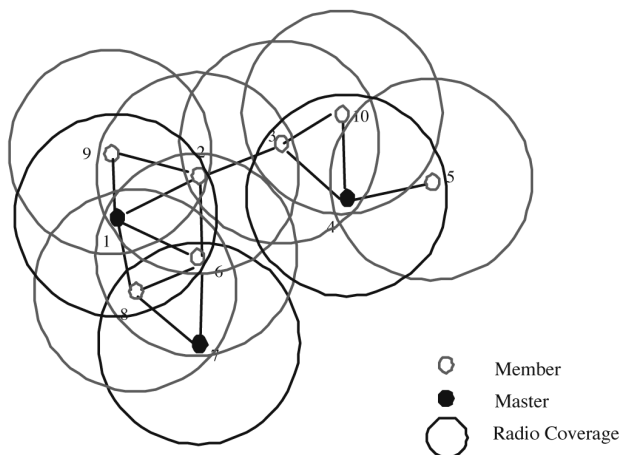


Fig. 4. Node Grouping.

then it becomes an island node.

- 3) If node A becomes an island node, then it tries to join another group. Node A first finds all the masters of its neighbors using its received RHELLO messages, then, it selects one master among them by sending a message RM to ask if it can join the group.
- 4) If the master receiving node A's RM accepts the request of node A, then it answers an agreement message RRM to make node A become its member.

The grouping of the network that has 10 nodes is shown in Fig. 4. The nodes are divided into 3 groups. The master of group 1 is node 1, and the members of it include node 1, 2, 6, 8 and 9. The master of group 4 is 4, and its members are 3, 4, 5 and 10. Node 7 is the master and the only member in group 7 because nodes 6 and 8 have joined group 1. The gateway node is node 6 or 8 between group1 and group7; the gateway nodes are node 2 and 3 between group1 and group4. In Fig. 4, it shows that the distance between two masters of two adjacent groups is at most 3 hops.

IV. DYNAMIC GROUP-CACHING STRATEGY

Caching technique has been widely used in distributed database systems. The servers connect with each other fixedly and persistently. The topology of a distributed database system seldom changes. However, in mobile computing environment, the topology changes frequently because nodes move frequently. The communication among nodes is not reliable, which makes traditional catching techniques not suitable for mobile database systems. Considering the characteristics of mobile computing environment, researchers have made an intensive study of the technology of caching, and proposed some caching strategies about mobile database systems [1], [12].

The same as in traditional distributed databases, caching is an efficient way to speed up query processing for mobile databases. Existing caching strategies are mainly based on BSs in wireless networks and not applicable for MANETs. Multi-hop routing in a MANET makes the distance (the number of hops) between two nodes the main factor of data transmission

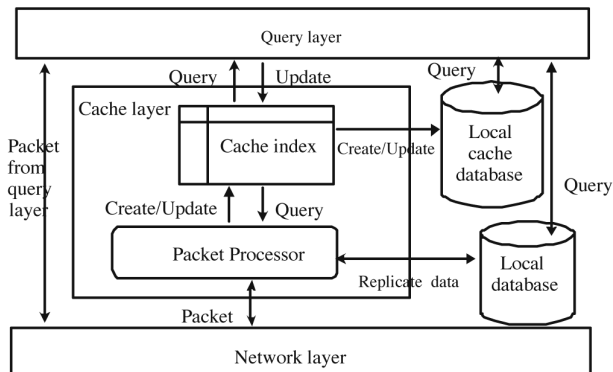


Fig. 5. Logic Frame of the Cache Layer.

cost. In distributed mobile databases, it is common to query the database at many nodes. For the limited cache space at a query node, merely caching data at the query node would decrease cache usage ratio. We present a group-caching strategy, in which data is cached in a group. The query response time is reduced by shorten the transmission distance. At the same time, all the nodes in a group could use the cached data and the cache usage ratio is increased.

Section 2 introduces the architecture of our distributed mobile database system. The cache layer of the system is in charge of data caching, which is shown in Fig. 5.

The dynamic group-caching strategy caches and distributes data items based on the probability they are accessed by nodes or groups and the data valid time (the time up to the next update of database). All the nodes in a group share the group cache. The cached items are distributed among different members in the group. The caching strategy updates the data in cache and maintains the index periodically to keep it in consistency with the original data.

In this strategy, cache index is divided into two levels, the node cache index and group cache index. The node cache index and the group cache index are defined as follows.

Definition 4.1: node cache index is a cached index maintained by a node. A segment in a database is the minimal cached unit, called cached item. It records the cumulative frequency of the segments accessed by the node, and whether the segments are cached at the node.

Definition 4.2: group cache index is a cached index maintained by the master of a group. It records the nodes in which the cached item located, the data valid time, the total frequency of the data accessed by the nodes in the group.

Using dynamic group-caching strategy, every group of nodes caches the data from outside of the group which is accessed frequently. It sets a high priority on the data whose valid time is longer. We use PT [10] to represent the possibility that a data item is cached in a group.

The definition of PT is as follows:

$$PT_{ij} = P_{ij} \cdot \tau_j = P_{ij} \cdot (T_j - t_j)$$

where P_{ij} is the frequency that node i requests for item j in a time unit. τ_j is the remaining time for the next update of data

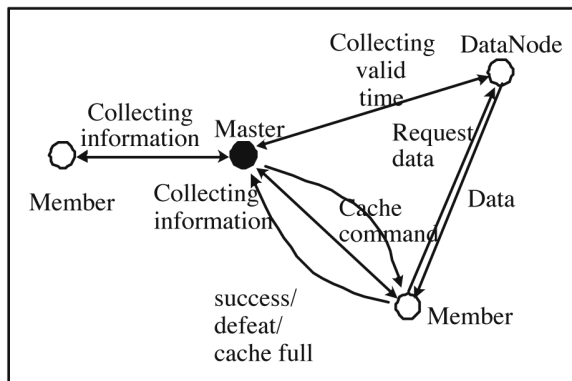


Fig. 6. Dynamic Group Cache Strategy.

item j . T_j is the current time, and t_j is the time when data will be updated next time. PT evaluates the frequency that a data item is accessed by a node in its valid time.

Definition 4.3: The group PT is defined as follows:

$$PT_{groupj} = \sum_{i=1}^n P_{ij} \cdot \tau_j$$

As shown in Fig. 6, the dynamic group-caching strategy consists of four phases: eliminating the invalid cached items, collecting information, assigning the cached items and replacing the items in cache.

- (1) Eliminating the invalid cached items. When nodes move, the members in a group might change, which causes the change of the group cache index. The master needs to delete the invalid items to maintain the group cache index. The valid time of all the data items in the group cache index is subtracted by one update cycle. The master deletes all the cached indices of the items whose remaining time are less than one update cycle, because they would be invalid before the next location update. At the same time, the master sends a message to the nodes whose indices are deleted, and notifies that node to eliminate the corresponding items from its local cache index and the database.
- (2) Collecting information. The master collects the accessing information of each item outside this group accessed by the nodes in the group, creates a new or updates the old group cache index. For the data segment that is not cached in the group cache index currently, the valid time of the segment is obtained from the node where it is.
- (3) Assigning the cached items. After the master collects the information, it deletes the indexed items whose valid time is less than one update cycle, and sorts the items in the group cache index by PT in descending order. If a cached item is not in the group, it is obtained from some remote data nodes, and cached at the member node where the item is accessed most frequently. If it is unsuccessful to cache the data item or the cache of the node is full, select the node whose data access frequency is inferior to this node, and so on.

- (4) Replacing the cached items. If the caches of all the members are full and there are cached items whose PTs are less than that of the current items, the cached index of the item with the minimal PT should be deleted from the group and the corresponding data item should also be deleted from the member in the group. After the cache is released, it is assigned to the current item.

The dynamic group-caching strategy determines the cache order of data items according to the information of the items accessed by the group. It also decides which node an item should be distributed to based on the frequency the item is accessed by the members. The strategy considers not only the frequency of data item accessed by the group, but also that accessed by each node in the group. Valid time is mainly considered by the strategy. The items with long valid time will be cached in priority. It improves the usability of the cache, and avoids the frequent replacement of the cached items.

V. GROUP-CACHING BASED QUERY PROCESSING

To introduce the query processing algorithm based on the group-caching strategy, we first take a single-table query on one node as example. The multi-table query processing on many nodes can be derived from this algorithm. A query with single-table on one node is expressed as follows:

```
SELECT Node.Table.Attr1, Node.Table.Attr2,.....
FROM Node.Table
WHERE Condition
```

After receiving a query statement from a user, the query node analyzes the query statement and constructs the query plan, which is submitted to the local or remote node to execute. With the group-caching strategy, the query results may be from the local data, the local cached data, the group cached data or the remote original data.

The single-node query algorithm based on group-caching is shown as follows:

1. IF QueryNode receives a query THEN
 - 1.1 IF it is a local query THEN search the local database;
 - 1.2 ELSE
 - 1.2.1 check the local cache index and acquire the data that satisfy the query condition.(matching set);
 - 1.2.2 deliver the query whose matching set is deleted to the group master;
2. IF Master receives the query THEN
 - 2.1 IF the data node belongs to the group THEN send the query to the node;
 - 2.2 ELSE search the group cache index, get the subset of items matching the query and the corresponding member nodes in the group;
 - 2.3 construct a subquery with the query set and the matching subset ; 2.4 send the subquery whose matching subset has been eliminated to corresponding member nodes;

3. *IF the data node receives the query THEN*
 - 3.1 *3.1 execute the query in local data or local cached data items;*
 - 3.2 *send the subquery result to the query node;*
4. *The Query node receives all the subquery results from other nodes and gets the final results.*

The query plan and its execution are shown in Fig. 7.

Let S be the size of the data in a query table. The distance from the data node to the query node is D hops. The cache hit rate is r . The speed of disk I/O is V_{disk} . The speed of wireless transmission is V_{net} .

Let $r \cdot S$ be the size of data that can get from the group cache for a query, $r \cdot S \cdot V_{disk}$ is the time to read the data from disk. Because the maximum distance between any two nodes in a group is 2 hops, the maximum time of transmitting the query results is $2r \cdot S \cdot V_{net}$ if the group cache items are hit. The size of query results obtained from a data node is $(1 - r) \cdot S$, the time of reading disk is $(1 - r) \cdot S \cdot V_{disk}$, and the time of transmission is $(1 - r) \cdot S \cdot V_{net} \cdot D$. The response time of the single-node query based on the group cache is as follows:

$$r \cdot S \cdot V_{disk} + 2r \cdot S \cdot V_{net} + (1 - r) \cdot S \cdot V_{disk} + (1 - r) \cdot S \cdot V_{net} \cdot D$$

Due to the speed of accessing disk is far more than the speed of wireless transmission, we omit the time for the disk I/O. Thus the response time of the query is shown as follows:

$$2r \cdot S \cdot V_{net} + (1 - r) \cdot S \cdot V_{net} \cdot D.$$

If we do not use the cache, all the query results must be obtained by data nodes. The response time of the query is $S \cdot V_{net} \cdot D$. Consequently, the ratio of the responding times with the cache to that without the cache is $1 - r + \frac{2r}{D}$.

VI. JOIN OPERATION AND THE COST MODEL

Mobile query processing [6] is an extension of distributed query processing. To support distributed query in mobile networks, the disconnection, bandwidth and the reliability must be considered. The mobile query optimization must consider the characteristics of the mobile network and adopt appropriate optimization strategies based on the status of the network.

In mobile distributed database system, the query cost is generally determined by the communication cost, the size of the transmitted data, the transmission distance and the ability of the nodes. The percentage of the bandwidth and the time occupied are the main costs [2].

Suppose that all the MHs have the same processing ability and wireless communication ability (so the effects of the processing ability and the communication ability are ignored). MHs may have different power. In a MANET, the communication cost is the main cost limited by the multi-hop routing, the bandwidth and the communication ability. The query cost is determined as following:

query cost Q = the size of the data $S \times$ the transmission distance $D +$ local query cost C ;

S : the number of the tuples $S \times$ the number of the bytes of a tuple;

D : the number of the hops in transmission C : the query cost at a query node or a data node

Data transmission is mainly resulted from the join and union operations. So the optimization of the join operation is very important.

A typical query with two-table multi-attribute join on two different nodes is expressed as follows:

```
SELECT Node1.Table1.*, Node2.Table2.*,
Node1.Table1.Attr1, Node2.Table2.Attr2, ...
FROM Node1.Table1, Node2.Table2
WHERE Node1.Table1.Attr1 = Node2.Table2.Attr2
Operator1 Node1.Table1.Attr3 = Node2.Table2.Attr4
Operator1 Node1.Table1.Attr5 Operator2 Value1
Operator1 Node2.Table2.Attr6 Operator2 Value2 ...
```

where,

Operator1 \in { and, or }

Operator2 \in { <, >=, <=, =, != }

The size of the join can be evaluated as follows [9]:

$$T(T_1 \bowtie T_2) = \frac{T(T_1) \cdot T(T_2)}{\prod_{j=1}^n \max(V(T_1, A_j), V(T_2, A_j))}$$

where $T(R)$ is the number of the tuples in relation R , $V(R, Y)$ is the number of the different values of the attribute Y in R , and A_1, \dots, A_i are the join attributes.

A distributed query optimization is composed of two parts: distribution optimization and local optimization. The distribution optimization is usually more important than the local optimization. It is based on reducing the communication cost. The local optimization can be done using the centralized database optimization techniques. This paper combines the two methods. The query plan generation algorithm is described as follows.

Query plan:

1. QueryNode sends the query to DataNodes;
2. After the DataNodes accept the query:
 - 2.1 process the query in local database, select the WHERE conditions on local data, project on the result attributes and the join attributes which are related to local data only;
 - 2.2 calculate the distance from itself to the QueryNode(Own_D) and the size of the local subquery result(Own_S);
 - 2.3 Sends Own_D and the information about the subquery result(Info) to the other DataNode;
3. Data nodes accept the distance information Other_D and the subquery results information Info from the other DataNode;
 - 3.1 calculate the distance from itself to the other data node(Between_D), calculate the data size after join

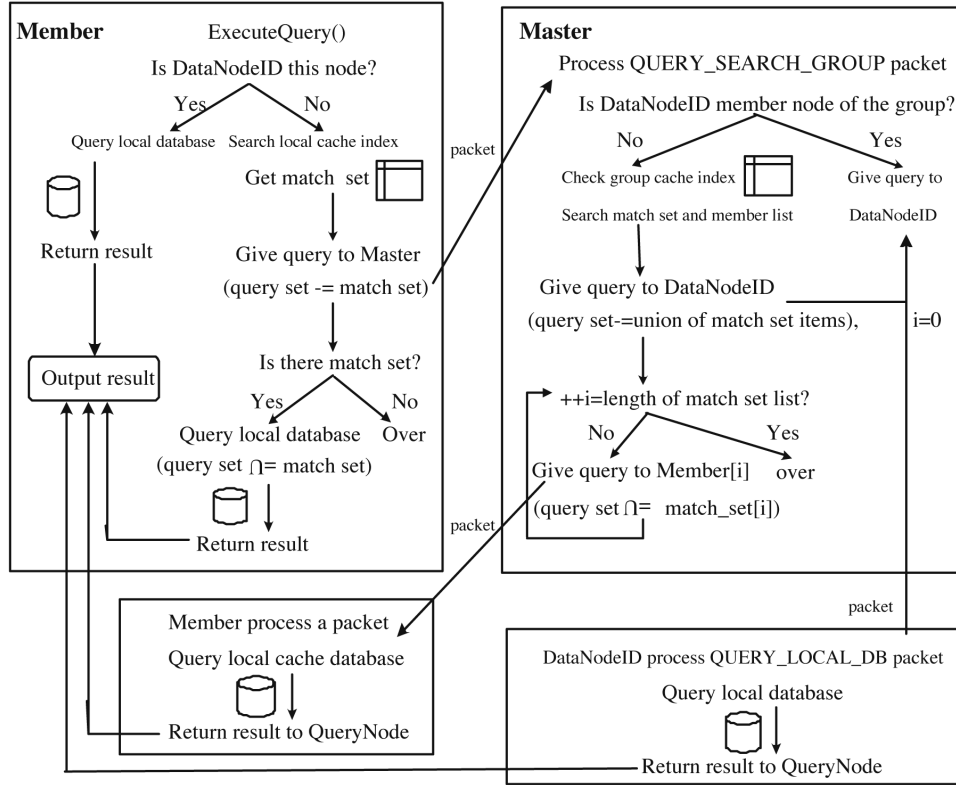


Fig. 7. Query Plan Based on Group Cache.

Join_S and the subquery result size of the other data node Other_S according to Info;

3.2 generate three subquery plans and the cost model:

P1: Both of the data nodes transmit the subquery results to QueryNode. Join is executed by QueryNode;

$$Q1 = Own_S * Own_D + Other_S * Other_D$$

P2: Local node transmits the subquery results to the other data node. The other data node executes the join and returns the join results to QueryNode;

$$Q2 = Own_S * Between_D + Join_S * Other_D$$

P3: The other data node transmits the subquery results to local node. Local node executes the join and returns the join results to QueryNode;

$$Q3 = Other_S * Between_D + Join_S * Own_D$$

if $Q1$ is the smallest, then P1 is selected

Transmit the subquery results to QueryNode;

if $Q2$ is the smallest, then P2 is selected;

Transmit the subquery results to the other data node;

if $Q3$ is the smallest, then P3 is selected;

Waiting to accept the subquery results from the other data node;

4. If the DataNode accepts the subquery results from the other data node, then

4.1 join local subquery results with the accepted results based on join conditions and the result attributes;

4.2 package the join results and return it to QueryNode.

5. If QueryNode accepts the join result package from DataNode, then output the results.

6. If QueryNode accepts two subquery results from both data nodes, then join the two subquery results on the join conditions and the result attributes and output the join results.

In the algorithm, QueryNode is the node that sends a query and DataNode is the node that executes the subquery.

The execution process of the query plan is illustrated in Figure 8. Besides, when both P2 and P3 are failure and the subqueries in both of the data nodes have finished, P1 is executed.

We analyze the cost of the two-node join algorithm based on the two-table single-attribute equal join of the two nodes.

Because the size of the subquery command from the QueryNode and the size of the subquery results transmitted between the two data nodes are small, the time to transmit the information among the nodes is omitted.

S_1 and S_2 are the sizes of the table T_1 at node 1 and the table T_2 at node 2. V_{disk} is the I/O speed of the local disk. V_{net} is the speed of the transmission. A is the join attribute. D_1 and D_2 are the distances from QueryNode to node 1 and node 2 respectively. The distance from node 1 to node 2 is D_b .

The time for reading/writing T_1 at node 1 and T_2 at node 2 is $S_1 \cdot V_{disk}$ and $S_2 \cdot V_{disk}$. The time for transmitting T_1 and T_2 to QueryNode is $S_1 \cdot V_{net} \cdot D_1$ and $S_2 \cdot V_{net} \cdot D_2$. The size of the join result of T_1 and T_2 is:

$$T(T_1 \bowtie T_2) = T(T_1) \cdot T(T_2) / \max(V(T_1, A), V(T_2, A))$$

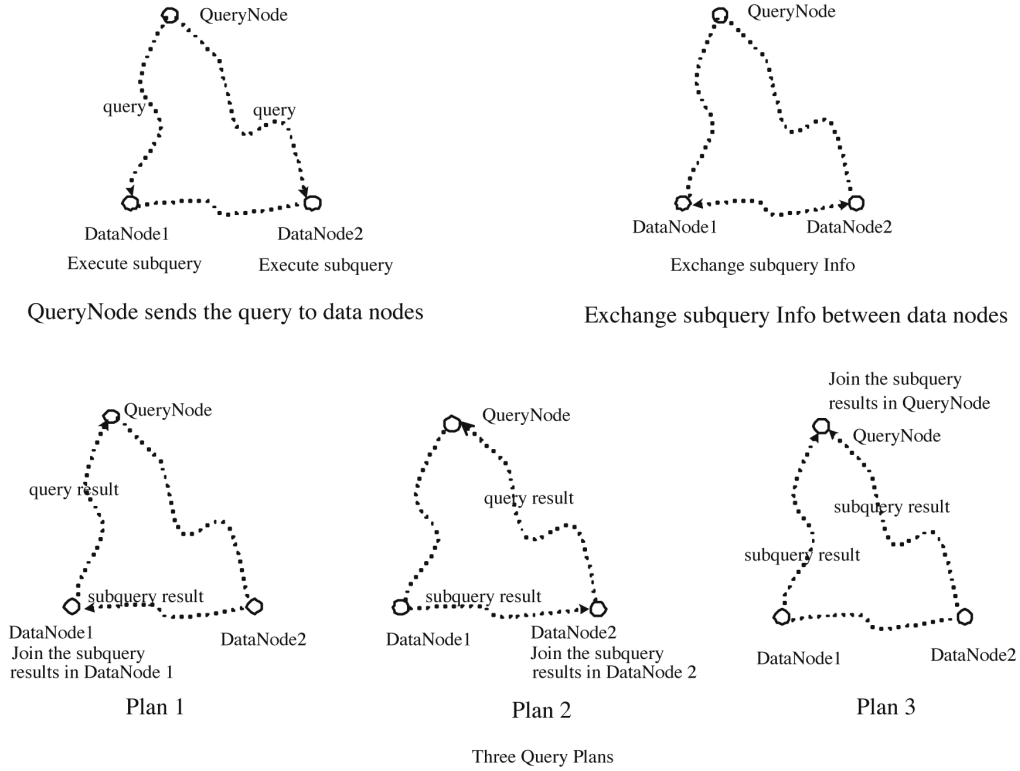


Fig. 8. The Execution of Query Plans.

The size of the join result of T_1 and T_2 is:

$$S_{join} = S_1 \cdot S_2 / \max(V(T_1, A), V(T_2, A)).$$

The time for the disk I/O of the join result of T_1 and T_2 is $S_{join} \cdot V_{disk}$. The transmission time is $S_{join} \cdot V_{net} \cdot D_1$ or $S_{join} \cdot V_{net} \cdot D_2$.

If the join operation is executed in QueryNode, the query response time is

$$T_q = S_1 \cdot V_{disk} + S_2 \cdot V_{disk} + S_1 \cdot V_{net} \cdot D_1 + S_2 \cdot V_{net} \cdot D_2 + S_{join} \cdot V_{disk}$$

If the join operation is executed at node 1, the query response time is

$$T_{S1} = 2S_2 \cdot V_{disk} + S_2 \cdot V_{net} \cdot D_b + S_{join} \cdot V_{disk} + S_{join} \cdot V_{net} \cdot D_1$$

If the join operation is executed at node 2, the query response time is

$$T_{S2} = 2S_1 \cdot V_{disk} + S_1 \cdot V_{net} \cdot D_b + S_{join} \cdot V_{disk} + S_{join} \cdot V_{net} \cdot D_2$$

Because the speed of the disk I/O is much higher than the speed of the multi-hop transmission in a MANET, we omit the time for the disk I/O. The join algorithm selects the one that has the smallest cost among the three query plans. The query response time of the two-node join is:

$$\min(S_1 \cdot V_{net} \cdot D_1 + S_2 \cdot V_{net} \cdot D_2, S_2 \cdot V_{net} \cdot D_b + S_{join} \cdot V_{net} \cdot D_1, S_1 \cdot V_{net} \cdot D_b + S_{join} \cdot V_{net} \cdot D_2)$$

For centralized join, the join operation is executed by QueryNode. The query response time of the two-node join is:

$$S_1 \cdot V_{net} \cdot D_1 + S_2 \cdot V_{net} \cdot D_2.$$

The ratio of the two-node join proposed in this paper to the centralized two-node join is

$$\min(1, (S_2 \cdot D_b + S_{join} \cdot D_1) / (S_1 \cdot D_1 + S_2 \cdot D_2), (S_1 \cdot D_b + S_{join} \cdot D_2) / (S_1 \cdot D_1 + S_2 \cdot D_2))$$

VII. EXPERIMENTS AND PERFORMANCE ANALYSIS

In order to test the proposed algorithms, a simulation environment for MANET has been built. The number of the nodes in the MANET consisting of 20 nodes, and all nodes are distributed within an area of 1000×1000 , effective communication radius of all the nodes are 300 units of length.

A. Experiments of queries based on group caching strategy

We simulated the single-node query based on group-cache, direct-cache and non-cache to study their performance. Group-cache uses the algorithm proposed in section 5. Direct-cache directly caches the data at single node, but without group management.

In this experiment, the update cycle of a node position was 10 seconds, i.e. a node moves to a new place per 10s. Each node submitted a random query periodically and searched for the data among the random nodes in the networks. To make the caching easily, the range of each query was 10 percent

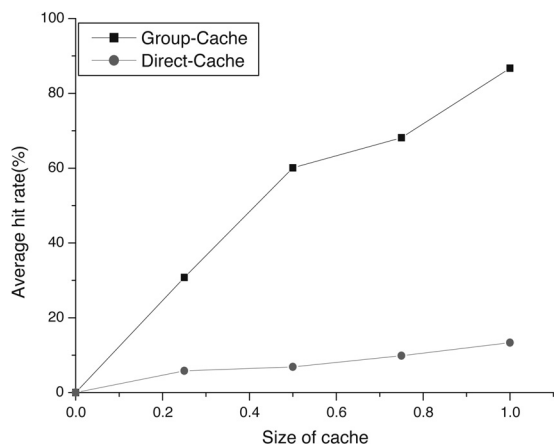


Fig. 9. Hit Rate of Cache.

of the data in the node. The size of the cache in a node was set to 0 percent, 25 percent, 50 percent, 75 percent and 100 percent of the querying range respectively. Figure 9 shows the hit rate of the group-cache and the direct-cache with the single-node query algorithm. Figure 10 shows the comparison of the response time between the query with group-cache, direct-cache and non-cache.

As can be seen, Fig. 9 shows the hit rate of the group-cache and direct-cache increases correspondingly to the size of the cache. But the hit rate of group-cache is distinctly higher than that of direct-cache. In the group-cache, not only because coverage rate of the cache which makes more data items cached in the group is increased, but also because the index management is set to the cache of nodes in the group. Thus, if only the query arrives at the group, it can quickly match the data cached in the group to obtain the whole or partial query results, increasing the number of query results and the hit rate. In the direct-cache, the query can hit if only it arrives at the node cached, otherwise it can not use the data cached. So the hit rate is low.

Figure 10 shows that the response time using the group-cache is less than that of direct-cache and non-cache. Because many data items relating to the query are cached in a group, these data could be shared among the nodes to enhance the query efficiency. On the other hand, due to the higher hit rate of group-cache, many queries can obtain the query results (or partial results) on the nodes cached to decrease greatly the frequency of obtaining directly the query results to the origin nodes. So using group-cache not only reduces effectively the amount of data transmission among the nodes, but also decreases the average response time of a query. In direct-cache, because of the lower hit rate of caching and less query results that can obtain on the data cached, the query frequently visits the origin data nodes. Thus the response time of a query using direct-cache is longer than that using group-cache, but shorter than that using non-cache. With an increase of the cache size in a node, the hit rate of group-cache and direct-cache increase gradually, and the query response of group-cache and direct-cache time is shortened gradually.

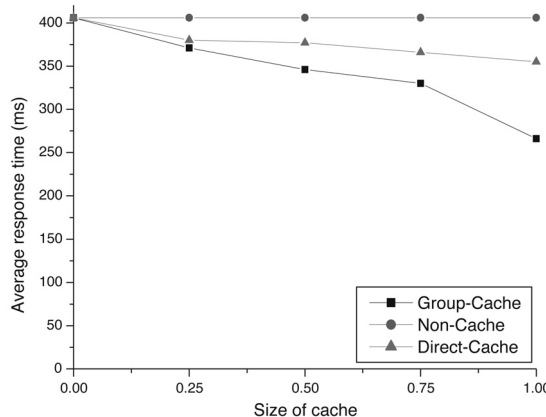


Fig. 10. Average Response Time of Single-node Query.

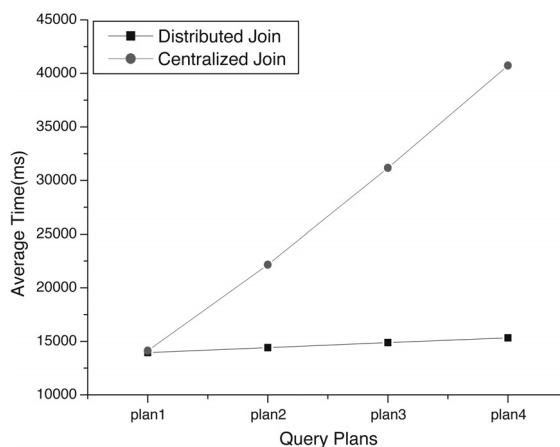


Fig. 11. Execution Time of the Two Join Methods.

B. Join query experiments on two nodes

The experiments for testing the distributed join algorithm proposed in this paper and the centralized join were conducted. Distributed join used the algorithm proposed in section 6. Centralized join first filtered on the two data nodes, than sent the sub-query results to QueryNode , and next joined the results on the query nodes using the subquery plan P3.

In these experiments, the update cycle of a node position was 5 seconds. Several nodes were selected to submit a random join-query periodically. Four query plans were tested to study their performance. Every plan used different data sets. In all the four plans, the size of R1 is 10000 tuples; the size of R2 ranges from 10000 to 40000 tuples, that is, plan1 is 10000 tuples, and plan2 is 20000 tuples and so on. The experiment results are shown in Figure 11.

Figure 11 shows that as the data amount increases, the respond time for the distributed join increases more slowly while the time for the centralized join increases more rapidly. In a MANET, the bandwidth of wireless communication is limited and nodes communicate by multi-hops, so the communication cost is the main cost of a query. Based on the characteristics above, different query plans are selected for different situations by the algorithm proposed in section 6. That is, the main optimization aim is to reduce the communication cost among

nodes. In the distributed join, the algorithm first do filtering on the two data nodes, then calculates the cost of different plans and finally executes the query using the plan whose cost is the minimum. Hence the distributed join can minimize the amount of communication among nodes, decreasing obviously the respond time. The centralized join is just the special case of distributed join (Plan 3 in Fig. 8). In this case, if the selectivity on one or two of the two data nodes is lower, then more data need be sent to QueryNode. So the response time is distinctly greater than that of distributed join

VIII. CONCLUSION

With the development of the mobile computers and the wireless communication techniques and the decreasing of the hardware cost, the applications of the mobile database have become more and more popular. Thus the demand for the research of database techniques in mobile networks has been growing rapidly. This paper focuses on the group-caching strategy, the group caching based query processing, the join operation, and the query optimization of join in mobile distributed databases in a MANET. The cost model of the join operation and the query plan generating algorithm are proposed. The group caching and the join strategy proposed in this paper are suitable for the characteristics of the MANET such as mobility, equity, multi-hops... It greatly reduces the amount of the communication and the query execution time.

REFERENCES

- [1] M. H. Dunham and A. Helal. Mobile Computing and Databases: Anything new? *SIGMOD Record*, 24(4), 1995.
- [2] H.-E. Kottkamp and O. Zukunft. Location-aware query processing in mobile database systems. *Proceedings of the 1998 ACM Symposium on Applied Computing*, 1998.
- [3] F. Perich, S. Avancha, A. Joshi and Y. Yesha. Technical report, Query Routing and Processing in Mobile Ad-hoc Wireless networkss, UMBC, October 2001.
- [4] L. D. Fife and L. Gruenwald. Research Issues for Data Communication in Mobile Ad-Hoc Network Database Systems. *SIGMOD Record*, 32(2), June 2003.
- [5] D. Barbará. Mobile Computing and Databases-A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1), January/February 1999.
- [6] H. Gökmen Gök and Ö. Ulusoy. Transmission of continuous query results in mobile computing systems. *Information Sciences*, 125(1-4), 2000.
- [7] S. K. Madria, M. K. Mohania and J. F. Roddick. A Query Processing Model for Mobile omputing using Concept Hierarchies and Summary Databases. *Proc. Foundations of Database Organisation, FODO'98*, Kobe, Japan.
- [8] T. Hara. Replica Allocation in Ad Hoc Networks with Periodic Data Update. *Proceedings of 28th VLDB Conference*, Hong Kong, China, 2002.
- [9] H. Garcia-Molina, J. D. Ullman and J. Widom, *Database System Implementation*. Prentice Hall, 2000.
- [10] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA. February 1999, pp. 90-100.
- [11] K.-L. Wu, P. S. Yu and M.-S. Chen. Energy-Efficient Caching for Wireless Mobile Computing. *Proceedings of the 12th International Conference on Data Engineering*, 1996, pp. 336-343.
- [12] S. Khurana, A. Kahol, S. K. S. Gupta and P. K. Srimani. An Efficient Cache Maintenance Scheme for Mobile Environment. *Proceedings of the 20th International Conference on Distributed Computing Systems*, Taipei, Taiwan, April 2000.

Jinbao Li received the BSc degree in electron engineering from Heilongjiang University, China, and the MS degree in computer science from Heilongjiang University, China. He is currently a PhD candidate in the Department of Computer Science at Harbin Institute of Technology, China. His research interests include sensor networks and ad hoc mobile wireless network. He has published more than 20 technical papers in refereed journals and conference proceedings.

Yingshu Li received the BSc degree in computer science from Beijing Institute of Technology, China, and the MS degree in computer science from University of Minnesota, USA. She is currently a PhD candidate in the Department of Computer Science at University of Minnesota. Her research interests include ad hoc mobile wireless network and sensor networks. She has published 16 technical papers in refereed journals and conference proceedings in the areas of wireless network.

My T. Thai received the BSc and MS degrees in computer science from University of Minnesota, USA. She is currently a PhD candidate in the Department of Computer Science at University of Minnesota. Her research interests include ad hoc mobile wireless network. She has published more than 5 papers in refereed journals and conference proceedings.

Jianzhong Li is a professor and the chairman of the Department of Computer Science and Engineering at the Harbin Institute of Technology, China. He worked in the University of California at Berkeley as a visiting scholar in 1985. From 1986 to 1987 and from 1992 to 1993, he was a staff scientist in the Information Research Group at Lawrence Berkeley National Laboratory, Berkeley, USA. He has also been a visiting professor at the University of Minnesota at Minneapolis, Minnesota, USA, from 1991 to 1992 and from 1998 to 1999. His current research interests include data warehousing, data mining, XML databases, bioinformatics, and sensor network data management systems. He has authored three books, including *Parallel Database Systems*, *Principle of Database Systems*, and *Digital Library*, and published more than 200 technical papers in refereed journals and conference proceedings in the areas of parallel databases, science and statistical databases, XML databases, sensor network data management systems, data mining and data warehousing. He has delivered a number of invited presentations and participated in panel discussions on these topics. His professional activities have included service on various program committees, and he has refereed papers for varied journals and proceedings. He is a member of the IEEE Computer Society and a member of the ACM.