Chapter 3

A Quorum-Based Scheme for Distributed Location Tracking

3.1 The System Model

Several existing systems for mobile computing networks assume the existence of a cellular system of mobile nodes and stationary nodes. In this chapter, the cellular system is modeled as a geographical area, which consists of many hexagonal cells. A fixed base station, called the *mobile support station* (MSS), supports each cell. The mobile support station is static and connected through a dedicated wire-line link to an existing wired backbone. The mobile node, referred to as the *mobile host* (MH), is a part of one and only one cell at a time. Each mobile host can only communicate through the MSS of a cell with any particular node, whose position has been located. Communication between MH and MSS occurs through radio waves or infrared waves, which are wireless. A *registration area* (RA) consists of several cells. Each registration area has one *location server* (LS) that maintains the location information of the mobile hosts in the RA [27]. All the location servers form a distributed location database in the mobile computing networks (Fig. 3-1).

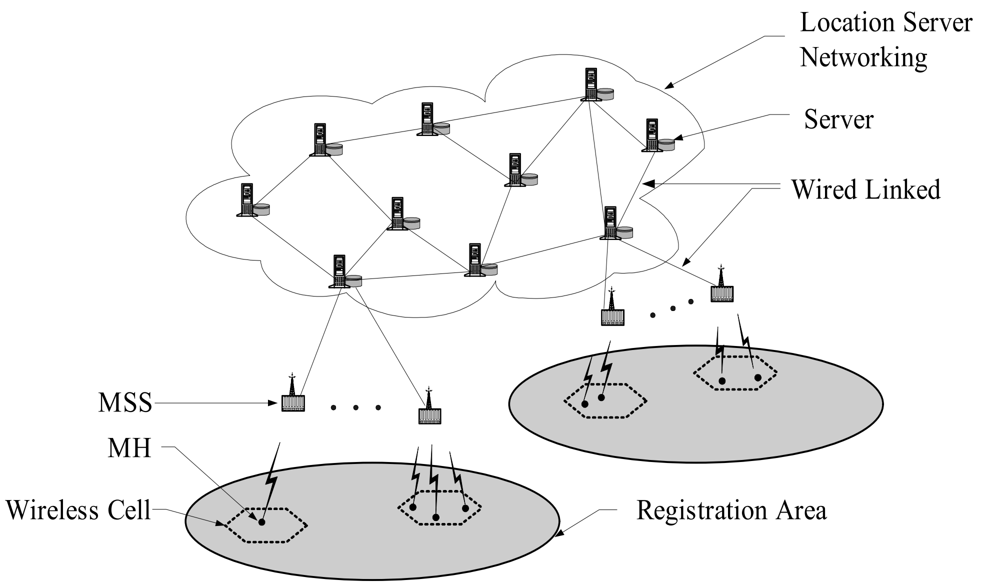

Fig. 3-1. Logical view of a mobile network with distributed location database.

3.2 Design Approach

In cellular mobile systems, location management is achieved by querying and updating. A query occurs when a host wants to communicate with another mobile host whose location is unexpected, and an update occurs when a mobile host changes its location. In a quorum-based scheme, the location information of a mobile host in an RA is stored in the local LS and replicated at all the LSs in the selected quorum. To extract the information from other LSs, the quorum of the query server and the quorum of the update server must be joint. Since Theorem 1 presents the intersection property of U-quorum and Q-quorum, the U-Set and Q-Set defined in Definition 4, a quorum-based scheme, can be used in location management.

In the following, we use the method of random selection [38] of quorums and timestamps along with our *LegRing* location management scheme defined in

Definition 4 to implement the location update and query algorithms.

Location Update: When a mobile host *h* moves from one cell to another, its location information has to be updated. Therefore, the following steps for update procedure are performed:

Step 1. The UPDATE message associated with the timestamp is stored into the cache of local LS and sent to all the LSs in the randomly selected quorum from the U-set.

Step 2. Upon receiving the UPDATE message, the LSs add or overwrite the new location information received to their caches.

Location Query: When a mobile host *h* wishes to communicate with another host *h*' whose location is unknown, the query procedure is invoked with the following steps:

Step 1. First, host *h* queries the location information of *h'* from its local LS. If the information is found in LS's cache, the corresponding MSS is probed to determine if *h'* is still in the cell. If so, the call is connected, and then the query procedure is stopped. Otherwise, the local LS clears the location information of host *h'* and then goes to step 2.

Step 2. The procedure randomly selects a quorum from Q-set and sends a QUERY message to all the LSs in the quorum for the location information of *h'*.

Step 3. When it receives a QUERY for information, the LS, which has a copy of the queried information, sends a REPLY containing the timestamp associated with the location information. Otherwise, the LS sends a NULL reply.

Step 4. When all the REPLY messages from all the LSs in the quorum are received, the procedure selects the location information with the latest timestamp, stores it in the local LS's cache, and returns the information to the MSS of mobile host *h*. According to the location information, the call to host *h'* can be connected.

Consider an *N-LegRing* system with 21 location servers. According to Definition 4, the U-set and Q-set are constructed as follows:

U-set={{0,1,2,3,4},{1,2,3,4,5},{2,3,4,5,6},{3,4,5,6,7},{4,5,6,7,8,},{5,6,7,8,9},{6,7,8,9,10},{7,8,9,10,11},{8,9,10,11,12},{9,10,11,12,13},{10,11,12,13,14},{11,12,13,14,15},{12,13,14,15,16},{13,14,15,16,17},{14,15,16,17,18},{15,16,17,18,19},{16,17,18,19,20},{17,18,19,20,0},{18,19,20,0,1},{19,20,0,1,2},{20,0,1,2,3}};

Q-set={{0,5,10,15,20},{1,6,11,16,0},{2,7,12,17,1},{3,8,13,18,2},{4,9,14,19,3},{5,10,15,20,4},{6,11,16,0,5},{7,12,17,1,6},{8,13,18,2,7},{9,14,19,3,8},{10,15,20,4,9},{11,16,0,5,10},{12,17,1,6,11},{13,18,2,7,12},{14,19,3,8,13},{15,20,4,9,14},{16,0,5,10,15},{17,1,6,11,16},{18,2,7,12,17},{19,3,8,13,18},{20,4,9,14,19}}.

If a mobile host *h* moves from one cell to another, new location information should be stored in the (new) local LS, for example, server 4, and all the LSs, for example, servers 19,20,0,1,and 2, in the randomly selected U-quorum. When a mobile host wants to communicate with host *h* whose location information is not in the cache of the local LS, for example, server 17, it can acquire the information from the other LSs, for example, servers 1,6,11,16 and 0, in the randomly selected

Q-quorum. Since quorum {19,20,0,1,2} and {1,6,11,16,0} have common servers, 0 and 1, the newest location information of host *h* can be extracted from these servers.

3.3 Fault-Tolerant and Load-Balanced Approach

The responsibility for location tracking using a deterministic quorum selection approach in which a quorum is initially selected by a procedure is not guaranteed to be shared equally by all the location servers. Hence, load unbalance arises when the deterministic quorum-based approach is used, which selects the quorum based on the geographical location of the mobile host. This occurs for the following two major reasons mentioned in [26]. First, a significant fraction of the mobile hosts may quite often be concentrated in a very small area, while there may be few mobile hosts in the remaining area of the system. Second, even if all the mobile hosts are evenly distributed across the system, some hot mobile hosts may be queried more often, thus increasing the loads of the location servers of some quorums. Because a random selection method is used in our quorum-based *LegRing* scheme, the implementation has a better chance of achieving load balance among the location servers since the selection of quorum is dynamic and each server bears even probabilities.

The quorum-based method is naturally fault tolerant if we use dynamic assignment of quorums. In the algorithm presented in section 3.2, the procedure can select another quorum if it does not receive all the REPLY information from all the servers of the selected quorum during a given period of time since some servers of the queried quorum may crash. In order to tolerate the server failures, we can rewrite some steps in the update and query algorithms presented in section 3.2 as follows:

In **Location Update,** step 2 is rewritten as two steps:

Step 2. Upon receiving the UPDATE message, the LSs add or overwrite the new location information received to their caches and send back the ACK message.

Step 3. If the procedure does not receive all the ACK messages from all the LSs in the quorum during a given period of time, then it randomly selects another quorum from U-set, sends the UPDATE message to all LSs in the new quorum again, and goes to step 2; otherwise, it stops.

In **Location Query**, step 4 is rewritten so as to achieve fault tolerance as follows:

Step 4. When all the REPLY messages from all the LSs in the quorum are received, the procedure selects the location information with the latest timestamp, stores it in the local LS's cache, and returns the information to the MSS of the mobile host *h*. According to the location information, the call to host *h'* can be connected. If the procedure has not received all the REPLY messages after a given period of time, then it goes back to step 2 (another loop to randomly reselect a new query quorum).

3.4 Comparison and Discussion

A comparison of our scheme with some other quorum-based schemes is shown in Table 1-1. In the triangular configuration, the number of nodes $n(n+1)/2$ is equal to N, where n is an integer. Similarly, in grid based scheme, the number of nodes $m*n$ is equal to N, where m and n are integers. In the other two schemes, iterative and *LegRing*, the number of nodes n is equal to N. Obviously, with this property, only the iterative scheme and our *LegRing* scheme are applicable to a system with any arbitrary numbers of nodes.

Since the quorum structure of Bae's triangular configuration is not symmetric, the loads are heavier in the corner nodes of the triangle topology. Therefore, the triangular configuration does not achieve load balance. Only the grid based scheme, iterative approach, and our *LegRing* scheme are symmetric. With dynamic hashing, the grid based scheme and iterative approach can achieve load balance. With random selection of quorums, our *LegRing* scheme can achieve load balance.

In the quorum-based location management scheme, when one node of the selected quorum is faulty, the update or query procedure will cause information retrieval failure. Dynamic hashing or random selection of the quorum can avoid information retrieval failure since they can reselect another quorum by re-executing the hashing or random function. However, in the triangular configuration, the selection of quorums according to rows or columns cannot be achieved to a fully tolerant scheme.

Obviously, our *LegRing* scheme has the smallest quorum size among all of the schemes listed in Table 3-1. Hence, control traffic and connection delay can be reduced.

	Quorum Size	**Symmetric**	**Load Balance**	**Fault Tolerance**	**Number of Nodes**
Triangular configuration [27]	$\sqrt{2N}$	No	No	Yes/Partial	$n(n+1)/2$
Dynamic hashing + Grid based scheme [21]	$2\sqrt{N}-1$	Yes	Yes	Yes	$m*n$
Dynamic hashing + Iterative approach [26]	$0.97N^{0.63}$	Yes	Yes	Yes	n
Random + *LegRing* scheme	$\sqrt{N}$	Yes	Yes	Yes	n

Table 3-1. Comparisons of quorum-based location management schemes.