Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems

Ravi Prakash
Dept. of Computer Science
University of Rochester
Rochester, NY 14627-0226.
prakash@cs.rochester.edu

Mukesh Singhal
Dept. of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210.
singhal@cis.ohio-state.edu

We present a new location management strategy that employs dynamic hashing and quorums. Location information of a mobile host (MH) is replicated at a subset of location servers. New location servers can be added to the system as the number of mobile hosts and/or location update and query rates increase. The proposed scheme requires at most two rounds of message multicasting for location update and query operations. This compares favorably with hierarchical schemes that require $O(\lg N)$ rounds of unicast messages, where N is the total number of location servers.

Using Quorums for Update and Query: If only the MH's identity is used to determine its location servers, then regardless of where the MH is located in the network, its location information will always be stored at the same server(s). Determining the location servers of an MH based solely on the cell in which that MH is present will lead to uneven distribution of responsibility.

Given an MSS_id, denoting the cell in which the mobile host is present, and an MH_id for that mobile host, we employ double hashing as follows:

$$h(MSS\_id, MH\_id) = (h'(MSS\_id) + MH\_id \times h''(MSS\_id)) \ mod \ m$$

where $[0,m-1]$ is the range of the hash function $h()$, $h'()$, and $h''()$. Functions $h'()$ and $h''()$ are uniformly distributed over the range $[0,m-1]$, and $h''(MSS\_id)$ is relatively prime to m. Given an (MSS_id, MH_id) pair, $h(MSS\_id, MH\_id)$ will be uniformly distributed over the range $[0, m-1]$. Each value of $h()$ is *a priori* mapped to a quorum of location servers.

Location Update: Let the old and new locations of an MH be old_MSS_id and new_MSS_id, respectively. *Purge_set* is the quorum for $h(old\_MSS\_id, MH\_id)$, representing servers storing outdated location information. The *inform_set* is the quorum for $h(new\_MSS\_id, MH\_id)$, representing servers that should store new location information. To update the location of an MH following operations are performed: (i) servers belonging to $purge\_set - inform\_set$ are asked to delete outdated location information, (ii) servers belonging to $inform\_set - purge\_set$ are asked to add present location information which is timestamped with the local clock value of the MH, (iii) servers belonging to $purge\_set \cap inform\_set$ are asked to replace old location information with new timestamped location information.

Location Query: Quorum for $h(MSS\_id, MH\_id)$ is the set of location servers that a node resident in the cell represented by MSS queries to locate the mobile host MH. If the queried server has location information, it is sent in the response. Otherwise, a NULL response is sent. Having received all the responses (NULL and non-NULL), location information with the latest timestamp is selected. As quorums for every (MSS, MH) pair intersect, latest location information of an MH can be accessed in one round of message exchange.

Load Balancing with Dynamic Hashing: As $h()$ is a uniformly distributed hash function, each quorum is expected to receive roughly equal number of location updates and queries. However, if a load imbalance does occur, dynamic hashing can be employed. We modify $h(MSS, MH)$ to be a hash function whose range of values can expand or contract, depending on the load in the system as follows: $h_i(a,b) = (h'(a) + b \times h''(a)) \ mod \ 2^i$, for $0 \leq i \leq limit$, for some sufficiently large constant $limit$. So, $h_i(a,b) = h_{i-1}(a,b)$, or $h_i(a,b) = h_{i-1}(a,b) + 2^{i-1}$. Therefore, if the quorum corresponding to a $h_{i-1}(a,b)$ was heavily loaded, the (MSS_id, MH_id) pairs that were all previously mapped to quorum for $h_{i-1}(a,b)$ are now split between two distinct quorums, corresponding to $h_{i-1}(a,b)$ and $h_{i-1}(a,b) + 2^{i-1}$. All other (MSS_id, VMH_id) pairs, that were not mapped to the heavily loaded quorum, are mapped to the same quorum as before.

Accuracy: This measures the likelihood of a query operation for an MH returning location information stored by the latest update operation corresponding to that MH. As quorums intersect, servers probed by the query message ($query\_set$) will have non-empty intersections with the latest $inform\_set$ and $purge\_set$.

During normal operation location queries return the latest location information. However, if a location query is made about an MH when the MH's location is being updated, there is a small probability of failure to return the latest location information in the following situation: at least one server in $purge\_set$ receives the query prior to deleting old information and all servers in $inform\_set$ receive query prior to the update.

Overheads: During location update and query, messages are multicast to at most two quorums of location servers. Hence, the message complexity of each location operation is proportional to the quorum size: $O(\sqrt{N})$ if projective planes based scheme is used. If the *crumbling walls* based scheme is employed, some quorums can be as small as $\lg N - \lg\lg N$, where N is the total number of location servers in the system.

References

[1] R. Prakash, and M. Singhal. Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems. Technical Report 649, Computer Science Department, University of Rochester, February 1997.