MULTI TREE DATA BASE ARCHITECTURE FOR LOCATION TRACKING IN NEXT GENERATION MOBILE NETWORK

SAILAJA M, RAMYA K

Assistant Professor, of Computer Science, PVP Siddhartha Institute and Technology, Vijayawada,

PVP Siddhartha Institute and Technology, Vijaywada,

E-mail: ramya_123448@yahoo.com

ABSTRACT

The next-generation mobile network will support terminal mobility, personal mobility, and service provider portability, making global roaming seamless. A location-independent personal telecommunication number (PTN) scheme is conducive to implementing such a global mobile system. In this paper, firstly we propose multi tree database architecture consists of a number of database subsystems, each of which is a three-level t also proposes indexing schemes for each type of location databases and analyzes their efficiency and cost in terms of database access time and storage requirement. Tree structure and is connected to the others only through its root. Results have revealed that the proposed database architecture for location management can effectively support the anticipated high user density in the future mobile networks.

Keywords - *Database Architecture, location management, location tracking, mobile networks.*

1. INTRODUCTION

THE next-generation mobile network will be an integrated global system that provides heterogeneous services across network providers, network backbones, and geographical regions [1]. Global roaming is a basic service of the future mobile networks, where terminal mobility, personal mobility, and service provider portability must be supported. In a wireless network, a node (mobile-phone) will be present in a region and each region will have a MSS. M ss is mobile-switching-station or tower. Each mss will have up to date information of all the nodes under its control. Nodes will be continuously roaming i.e. it will change its location randomly. When ever a node leaves a region and enters another region, two region's mss will be updated. Each mss contains two databases namely HLR and VLR.

HLR is home-location-register which contains information about the nodes which are registered to operate in that area. VLR is visitor-location-register which contains location information about the nodes which are current in its area. The main aim of proposed concept is to provide minimum number of updates or evaluations (queries) when various service providers are going to be combined.

That is, there will be different nodes under different service provider under a same area. For example, in a region, there will be 1000 nodes for AIRTEL, 1000 nodes for AIRCEL and 1000 nodes for BSNL. AIRTEL, AIRCEL and BSNL are maintaining different towers to handle their calls. But in future, if all the service providers are going to be combined, then the number of users will be increasing tremendously. at that time, the paper tells that the HLR-VLR scheme is not sufficient. Proposed scheme contains three databases db0, db1 and db2. Db0 contains service profiles of all nodes globally. Db0 is the top-level and it contains pointers for each node to its db1, which in turn contains pointer to one of db2's where that node currently resides. Db2 contains copies of service profiles of nodes under it. So, db0 and db2 contains both index file and data file. Db1 contains only index i.e. it contains only pointers to db2. The paper tells that using the proposed architecture, the number of queries (evaluations) will be decreased. This is because since db2 contains data file also, it need not to access db0 every time whenever a node changes its location or a call is established. But in HLR-VLR scheme if more number of nodes resides in a region, then entire process will slow down because of slow time of location update procedures.

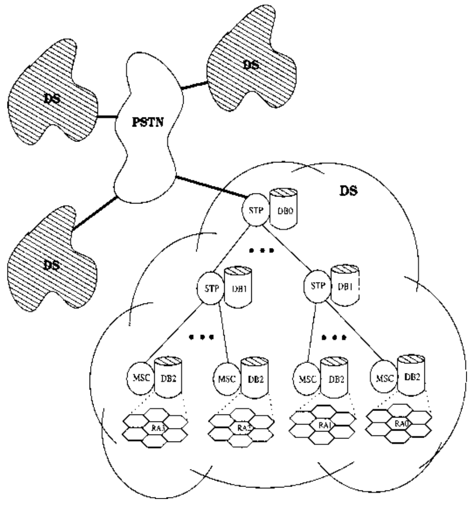

Fig. 1. **Proposed multi tree database architecture**

The proposed database architecture is motivated by the following.

1) A location-independent PTN provides a basis for global roaming in the next-generation mobile networks where terminal mobility, personal mobility, and service provider portability will be implemented. A mobile subscriber can retain its lifelong PTN regardless of its location and service provider.

2) The multi tree database architecture is much more robust than the one-root hierarchical architecture. In the proposed architecture, an MT's profile is stored in one of the root databases according to its current location. Thus, each root database only maintains a small portion of the user profiles in the global mobile system. The crash of one root database will not disrupt the operation of other root databases, and the recovery of the failed root database is much easier than in the one-root database architecture where all user profiles need to be recovered once the root is crashed.

3) The multi tree database architecture is scalable, which is crucial to support continuously increasing number of mobile subscriber's in future mobile networks. When the capacity of a root database is saturated, a new DS is readily added. More importantly, the end-to-end delay in location registration and call delivery will not increase due to such an expansion in the mobile network. On the other hand, with the one-root structure, when the capacity of the root or a high-level database is saturated, more levels of databases need to be added in order to reduce the burden on the root or high-level databases. This will increase the delays in location registration and call delivery.

4) The proposed multi tree database system is easy to expand and maintain in the multi operator environment of a global mobile system. With the multi tree architecture, each service provider can have its own DSs and it is straightforward for a service provider to expand its service coverage by adding new DSs. It is also easy to operate and manage a DS when the DS is

wholly owned by a single service provider. The one-root architecture, however, may not have such advantages.

5) No GTT is required in the proposed database architecture, where a signaling message is only sent from a database to another database in an adjacent level within the same sub tree or from a DB0 to another DB0. Since a message sender always contains the address of the receiver in its database, no GTT is required. This greatly simplifies the implementation of the proposed architecture.

In addition to the multi tree location database architecture, this paper also proposes indexing schemes for each type of location databases and analyzes their efficiency and cost in terms of database access time and storage requirement. The location registration and call delivery procedures based on the proposed database structure are also given. Analysis models are developed to study the service response time of each type of databases in the proposed multi tree architecture as well as the end-to-end delays incurred by the proposed location registration and call delivery procedures. The proposed architecture is compared with the one-root architecture as well as the HLR-VLR architecture in terms of the signaling loads due to location registration and call delivery. Numerical results have demonstrated that the proposed database architecture outperforms the one-root architecture and the HLR-VLR architecture, and can effectively cope with the anticipated high access rates to various location databases in future mobile

The remainder of this paper is organized as follows. Section II describes the proposed distributed database architecture for location tracking as well as the indices of the location databases. Section III presents the database searching strategies associated with the location update and call delivery procedures. And conclusions are given in Section IV.

2. MULTITREE DATABASE ARCHITECTURE FOR LOCATION TRACKING

A. **Multi tree location Database Architecture**

The proposed database architecture for location tracking is a multi tree structure, where each subsystem is a three-level architecture (Fig. 1), referred to as a *database subsystem* (DS) in this paper. Various DSs may represent networks operated possibly by different service providers. All these DSs are interconnected together via a fixed network, such as PSTN or ATM net- work, and communicate with each other only through their root databases. This architecture can support a multi operator environment which is expected in future mobile networks. In each DS, databases DB0 and DB2 may correspond to the HLR and the VLR in the two-level database system, respectively. Each DB2 may control an RA where a user can roam freely without triggering registrations. Each DB2 is co-located with an MSC, which performs call processing on origination or termination calls. A number of DB2s are grouped into one DB1 and severalDB1s are connected to a single DB0. Each DB1 and DB0 also needs a switch, called the STP that provides routing functionality for message exchange between various location databases. The DB0 maintains the service profile for each user currently residing in its service area, and maintains an entry for each user in the global mobile system. The entry contains either a pointer to another DB0 where the user is residing or a pointer to the user record that contains a pointer to the DB1 with which the user is currently associated. Each DB1 has an entry for every currently residing user, storing a pointer to the DB2 the user is currently visiting. Every DB2 has a copy of the service profiles of the users currently roaming within its area. With this architecture, the frequency of queries to the higher level databases is greatly reduced due to the locality of calling and mobility patterns.

However, when a call or a location update is not local, more databases—including the large centralized database DB0—need to be visited. This increases the end-to-end delays in call setup and location registration. In addition, as the number of mobile subscriber's increases, the access time of each database is increased, which also increases the end-to-end delays? To meet the delay demands in call setup and location registration, the number of database levels in a DS has to be limited. Moreover, to support a larger amount of mobile sub- scribers while keeping the end-to-end delays low, it is critical to reduce the access times to the databases. Thus, investigation into efficient database access indices for the location databases is as important as research into the overall location database architecture.

B. Two Efficient Database Indices

A database usually consists of two parts: an index file and a data file. The index file contains an access structure called index, which provides search paths for locating the records in the data file. The index determines the database access time, thereby being the critical component for improving database throughput. Efficient indices should be based on application characteristics such as the types of storage devices available, the affordable storage capacity, the types of queries required, the available keys, etc.

In this paper, we focus on the indices suitable for a variety f databases in mobile systems. There are two classes of indices: the disk-oriented index, such as the B -tree, and the memory-resident index, such as the AVL-tree and the T-tree. While the disk-oriented indices are designed primarily to minimize the number of disk block accesses and to minimize disk space, the memory-resident indices aim to reduce computation time while using as little memory as possible. For real-time applications, the memory-resident indices are preferred due to their much faster access times than the disk-resident indices. The indices can also be classified into the following two categories: the order-preserving indices and the randomizing indices. The primary order-preserving indices include arrays, B-trees, AVL-trees, T-trees, and direct files. The randomizing indices include various hashing indices. Essentially, the direct file is a special form of hashing indices

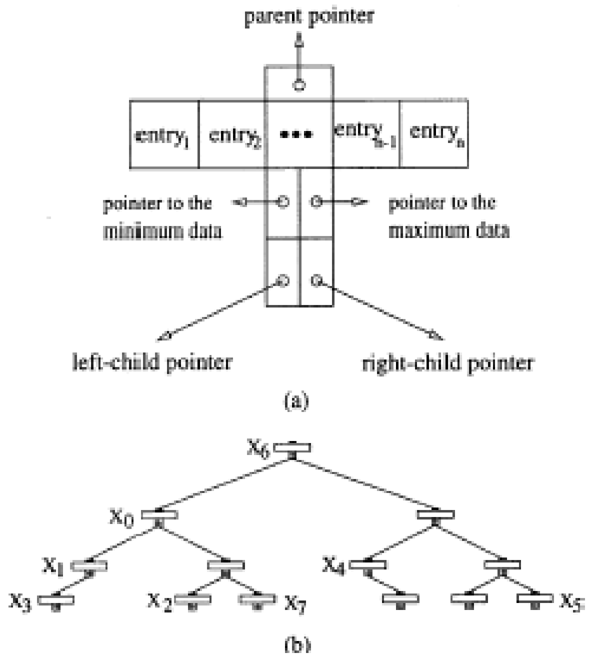

Fig: T-node. (b) T-tree.

We can call the direct file perfect hashing due to its collision-free property and use it in the DB0s due to its fast response time and easy implementation. The hashing indices have been applied in various computer and communications systems. For example, a hash function was used to balance the query load across multiple GTT servers by distributing users' PTN-to-HLR address mappings evenly among the GTT servers. In the peer-to-peer systems, hash-based techniques were used to map file names to their locations in the peer-to-peer systems while balancing the query load amongst all nodes. The hardest task of applying hashing techniques is to design efficient hash functions that can minimize collisions while keeping memory usage low. On the other hand, the order-preserving indices are much easier to implement and provide guaranteed upper bounds on the search time while keeping memory usage efficient. It has been shown that among the order-preserving indices-array, B-tree, AVL-tree, and T-tree, the T-tree provides the best overall performance for a mix of searches, inserts, and deletes at a relatively low storage cost. Inserts and deletes incurred by location update as well as searches required by call delivery in the DB1 and the DB2 make the T-tree suitable for these databases. On the contrary, the biggest drawback with the array is that data movement is for each update, thus the array seems only suitable for a read-only environment. The AVL-tree has poor storage utilization since each node stores only one data item while requiring two pointers and some other control information. As mentioned earlier, we also suggest that the memory-resident direct file be used as the index for large databases such as DB0, etc., due to its much faster speed than the other order-preserving indices.

T-Tree: The T-tree, which evolved from the AVL-tree and the B-tree, is a binary tree in which each node called T-node contains a number of data items, a parent pointer, a left-child pointer, a right-child pointer, and some other control information (Fig. 2). The T-tree is fast since it retains the intrinsic binary search nature of the AVL-tree. On the other hand, unlike the AVL-tree that holds only one data item in each node, the T-tree contains a number of data items in each node similar to the B-tree, thus having good storage utilization.

Direct File: In the direct file, there is a direct relationship between the record key and its storage location. The fastest searching method to access a direct file is direct addressing. The key value is used as a relative record number that can be translated into a hardware address by the system. When the direct file is memory resident, the hardware address is the memory address. One potential disadvantage of direct addressing is that space must be reserved for every possible key value, resulting in wasting large amounts of storage. However, when the number of possible key values is relatively close to the number of actual key values, direct addressing is very cost effective. Whenever access time is the vital criterion, even lower packing densities are acceptable.

3.LOCATION REGISTRATION AND CALLDELIVERY PROCEDURE

Location Registration and Call Delivery Procedures:

The location tracking procedures are described based on the proposed multi tree database architecture as well as the proposed database organizations. Location tracking consists of two procedures: the location registration procedure and the call delivery procedure. Location registration is the procedure through which a user reports its location to the network whenever the user enters a new location. As an incoming call arrives, the call delivery procedure is invoked to deliver the call to the user. For simplicity, in this paper, it is assumed that a DB2 only controls one RA. In real applications, a DB2 may control several RAs.

Location Registration Procedure:

With the previously defined file structures of DB0, DB1, and DB2 as well as the proposed multi tree location database architecture, the location update procedure in a global mobile system can be described as follows.

1) When a user enters a new RA, a registration request message is sent to the associated DB2 which in turn sends a registration request message to the DB1 controlling this area. If the user has no entry in this DB1, go to step 3; otherwise, go to step 2.

2) The fact that the user has an entry in this DB1 indicates that the new DB2 is within the same DB1 area as the old DB2. A pointer to the new DB2 replaces the old one in the user's entry in the DB1. No further query to the DB0 is needed. The DB1 sends a registration cancellation message to the old DB2, then go to step 8.

3) The fact that the user has no entry in this DB1 indicates that the user has moved to a new DB1 area. In the new DB1 an index entry is added to contain a pointer to the new DB2 of the user. An update request is also sent to the associated DB0.

4) The DB0 is checked to see if it contains the user's service profile. If no, this means that the user enters a new DS, then go to step 5a; otherwise, the DB0 updates the user's service profile to point to the new DB1 and sends a registration cancellation message to the old DB1, then go to step 7.

5) a) The new DB0 sends a query to the old DB0 to request the user's service profile.

b) The new DB0 stores the user's service profile and updates the service profile to point to the new DB1. A copy of the user's service profile is also sent to the new DB2.

6) a) The old DB0 sends the user's service profile to the new DB0.

b) The old DB0 updates the user's entry in the index file to point to the new DB0, and deletes the user service profile from its data file. A registration cancellation message is sent to the old DB1.

7) The old DB1 deletes the user's index entry, and sends a registration cancellation message to the old DB2.

8) If the old DB2 is in the same DS as the new DB2, a copy of the user's service profile is sent to the new DB2. The user's index entry as well as the user's service profile is removed from the old DB2.

9) After receiving the user's service profile, the new DB2 sets up an index entry for the user and create the user's service profile. The location registration procedure is completed.

Note that when a user changes its DS, with the preceding location registration procedure, only the old DB0 points to the new DB0 directly. All other DB0s (except for the new DB0) still point to the old DB0. The length of the forwarding pointer chains will increase as the user continues to change its DS. As a result, the end-to-end setup delay will increase for inter-DS calls. Compared to the single root structure, the proposed multi tree structure achieves its robustness, scalability, maintainability, etc., at the expense of the necessary of synchronizing the DB0s to contain the call setup delay as an MT changes its DS.

Call Delivery Procedure:

When an incoming call arrives, the call delivery procedure for the cal lee can be performed in the following steps:

1) When a call is detected in the caller's MSC, the caller's DB2 is checked to see if an index entry for the cal lee exists. If yes, go to step 5, and no further queries to the DB1 and the DB0 are required. Otherwise, a query is sent to the associated DB1, then go to step 2.

2) The DB1 examines if the cal lee has an entry in its index file. If yes, go to step 4, and no further query to the DB0 is required. Otherwise, a query is sent to the associated DB0, then go to step 3.

3) The DB0 examines if the cal lee is associated with one of its DB1s. If yes, the DB0 sends a routing address request message to the DB1, then go to step 4; otherwise, go to step 7.

4) The DB1 determines the cal lee's DB2 and sends a query to the DB2 to request the routing address.

5) The DB2 searches for the cal lee. If the cal lee is found, a TLDN is allocated to the cal lee and sent back to the calling MSC.

6) After receiving the TLDN, the calling MSC sets up a connection to the called MSC associated with the cal lee's current DB2. Then the call delivery process stops.

7) If the cal lee is residing in another DS, a query is sent to the associated DB0. The searching process is repeated from step 3.

It is worthwhile to point out that no GTT is required in the location registration and call delivery procedures based on the proposed database architecture. This will simplify the deployment of the proposed strategy while reducing the overall system cost.

4. CONCLUSION

Distributed multi tree database architecture has been proposed for location management in a global mobile system, where the location-independent PTNs are employed to support seamless global roaming. To support the anticipated large number of mobile users in the future mobile system, two efficient database access structures—the memory-resident direct file and the T-tree—were proposed to achieve high database throughput, so that the end-to-end delays in location registration and call delivery can meet the delay requirements in mobile networks.

The proposed database architecture is scalable, robust, and efficient. Compared to the existing two-level location database architecture, the proposed database architecture can support a much higher user density while reducing signaling load significantly. Compared to the one-root tree architecture, the proposed architecture provides better scalability and reliability while supporting a larger user population at a lower signaling cost. For performance evaluation, analysis model was developed. Numerical results have revealed that the proposed database architecture can effectively handle the anticipated high update and query rates to the location databases in future mobile networks. The proposed database access structures are also suitable for other large centralized databases in mobile networks, such as the authentication center and the equipment identity register.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments, which have significantly improved the quality of this paper.

REFERENCES

[1] I. F. Akyildiz, J. McNairy, J. S. M. Ho, H. Uzunalioglu, and W. Wang, “Mobility management in next-generation wireless systems,” *Proc. IEEE*, vol. 87, pp. 1347–1384, Aug. 1999.

[2] B . G. Clay brook, *File Management Techniques*. New York: Wiley, 1983.

[3] I.-R. Chen, T.-M. Chen, and C. Lee, “Agent-based forwarding strategies for reducing location management cost in mobile networks,” *ACM/Blitzer J. Mobile Newt. Applicants,* vol. 6, no. 2, pp. 105–115, 2001.

[4] S . Dolev, D. K. Pradhan, and J. L. Welch, “Modified tree structure for location management in mobile environments,” *Computer. Common.* vol. 19, no. 4, pp. 335–345, 1996.

[5] R . Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 2nd Ed. Menlo Park, CA: Addison-Wesley, 1994.

[6] J . S. M. Ho and I. F. Akyildiz, “Dynamic hierarchical database architecture for location management in PCS networks,” *IEEE/ACM Trans. Networking*, vol. 5, pp. 646–660, Oct. 1997.

—— [7], “Local anchor scheme for reducing signaling costs in personal communications networks,” *IEEE/ACM Trans. Networking*, vol. 4, pp. 709–725, Oct. 1996.

[8] R. Jain and Y.-B. Lin, "An auxiliary user location strategy employing forwarding pointers to reduce network impacts of PCS," *ACM-Baltzer J. Wireless Newts,* vol. 1, no. 2, pp. 197–210, July 1995.

[9] R. Jain, Y.-B. Lin, C. Lo, and S. Mohan, "A caching strategy to reduce network impacts of PCS," *IEEE J. Select. Areas Common,* vol. 12, pp. 1434–1444, Oct. 1994.

[10] R. Jain, S. Rajagopalan, and L. F. Chang, "Phone number portability for PCS systems with ATM backbones using distributed dynamic hashing," *IEEE J. Select. Areas Common,* vol. 15, pp. 96–105, Jan. 1997.

[11] L. Kleinrock, *Queueing Systems: Vol. I—Theory*. New York: Wiley, 1976.

[12] T. J. Lehman and M. J. Carey, "A study of index structures for main memory database management systems," in *Proc. 12th Int. Conf. Very Large Data Bases*, Aug. 1986, pp. 294–303.

[13] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architecture*. New York: Wiley, 2001.

[14] C. N. Lo and R. S. Wolff, "Estimated network database transaction volume to support wireless personal data communications application," in *Proc. IEEE Int. Conf. Communications*, May 1993, pp. 1257–1263. [15] A. D. Malyan, L. J. Ng, C. M. Leung, and R. W. Donaldson, "Network Architecture and signaling for wireless personal communications," *IEEE J. Select. Areas Common,* vol. 11, pp. 830–840, Aug. 1993.

[16] Z. Mao, "Location management strategies for personal communications services networks," Ph.D. Dissertation, Dept. Elect. Comput. Eng., Univ. Miami, Miami, FL, 2000.

[17] S. Mohan and R. Jain, "Two user location strategies for personal communications services," *IEEE Pers. Commun.*, pp. 42–50, First Quarter 1994.

[18] X. Qiu and V. O. K. Li, "Performance analysis of PCS mobility management database system," in *Proc. IEEE IC3N*, Sept. 1995, pp. 434–441.

[19] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A scalable con- tent-addressable network," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 161–172.

[20] N. Shivakumar, J. Jannink, and J. Widom, "Per-user profile replication in mobile environments: Algorithms, analysis, and simulation results," *ACM/Baltzer J. Mobile Newt. Applicant.* vol. 2, no. 2, pp. 129–140, Oct. 1997.

[21] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer- to-peer lookup service for Internet applications," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 149–160.

[22]E. D. Sykas and M. E. Theologou, "Numbering and addressing in IBCN for mobile communications," *Proc. IEEE*, vol. 79, pp. 230–240, Feb. 1991.

[23] J. Z. Wang, "A fully distributed location registration strategy for universal personal communication systems," *IEEE J. Select. Areas Common,* vol. 11, pp. 850–860, Aug. 1993.