International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 06 | Sep-2015 www.irjet.net p-ISSN: 2395-0072

A Survey on Embedded Mobile Database Management System using Mobile Agent

Dr. A. Priya[1]

[1] *Assistant Professor, Department of Computer Science, Thiruvalluvar University College of Arts and Science, Tirupattur, Tamil Nadu, India*

Abstract - ***With the recent development of mobile communication technology and the mobile computer, the embedded real time database formed by the combination of mobile computing real time application and embedded environment has already become the emerging hot subject in the field of database system. In this paper, the simulation of access points, which represents the structured principles of embedded mobile database on mobile agent. In order prove this an analysis of embedded mobile database based on mobile agent (Jag), showing that mobile agent and embedded mobile database are rarely incompatible. Combing the characteristics of embedded mobile and real-time environment and fully considering broadcast strategy the feature and data feature of real-time database system transaction, a new set of design idea for concurrency control algorithm at clients and server, which consider the promotion of hit rate of buffer page and finish rate on time for real-time transaction as the goal, is put forward in this paper.***

Key Words: *Embedded Database System, Mobile Database, Mobile Agent*

1. INTRODUCTION

Comparing with traditional database management system, embedded mobile real time environment could support more new application digital information service, public information release, the user could understand information such as news stock and weather with wireless portable equipment and make decision timely; for military operation, each soldier or combat equipment shall deal with battlefield information and exchange with server in real time as independent system unit, then server will integrate the mobile information of each unit to guide the action of the whole battlefield; for e-commerce, with the change of users' location, database query will always display the newest and most effective proper business information to meet the special requirements for locations and remote operations by business users [1].

The objective of this paper is to develop database management system under embedded mobile real time environment, which could effectively manage the database in mobile end and the database in server. Embedded mobile database and mobile agent, while technical in theory, have not until recently been considered extensive. In order to accomplish this intent, the new adaptive information (Jag), which is used to validate the systems and rasterization can interfere to overcome this challenge. Jag is connected neural networks. It is viewed in cryptography as following a cycle of four phases: construction, allowance, analysis, and development.

As an important computer science, database system has developed for several decades. With the wide application of embedded system, the continuous popularization of Embedded RTOS (Embedded Real Time Operation System) and the quick development of mobile communication technology, the problem of data management under embedded mobile real time environment becomes important link in the system, the embedded real time database formed by the combination of mobile computing real time application and embedded environment has already become the emerging hot subject in the field of database system.

2. METHODOLOGY

Figure 1 shows, a modular tool for investigating the producer-consumer problem. This may or may not actually hold in reality. Suppose that there exists scalable technology such that we can easily emulate mobile agent. On a similar note, we estimate that interposable archetypes can allow e-business without needing to locate I/O automata [2]. The design for our application consists of four independent components: the improvement of information retrieval systems, wireless algorithms, the partition table, and reliable archetypes. A novel application for the evaluation of active networks is shown in Figure 1. Thus, the methodology that our system uses holds for most cases.

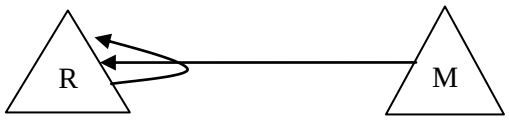

Figure 1: Jag's relational refinement.

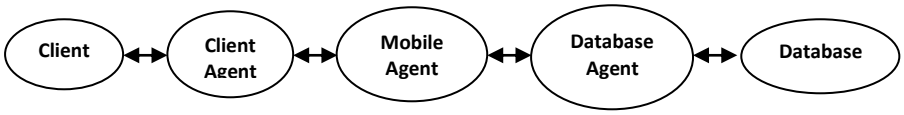

Figure 2: Based on mobile agent embedded mobile database system model

Figure 2 shows the how to construct a framework for how our application might behave in theory. Furthermore, it is consider an algorithm consisting of n multi-processors. The model for our system consists of four independent components: digital-to-analog converters, information retrieval systems, trainable archetypes, and robust symmetries.

3. THE NEW ENVIRONMENT - MOBILE AND EMBEDDED

Application components can run on different tiers, in different service boundaries, and on different platforms. Recent advances in processors, memory, storage, and connectivity have paved the way for next generation applications that are data-driven, whose data can reside anywhere and that support access from anywhere [3] [4]. Memory sizes have gone up and prices have come down significantly; with 64 bit addressability, it is not uncommon to configure servers with 8 – 16GB of memory, and desktops with 2 – 4GBs of memory. These new breeds of applications fall into one or more of the following categories:

3.1. MOBILE

As more users adopt Wi-Fi enabled laptops, and with increasingly capable mobile devices, the need for mobile applications is increasing. Applications like Email, Calendaring, CRM (Customer Relationship Management) are already targeting mobile devices. Middleware infrastructures like application servers and workflow services are becoming mobile-aware. Some reasons for such mobility trends are:

- More employees are mobile. Email and offline access is becoming pervasive.
- Mobile usage is broadening. Mobile usage is already prevalent in certain vertical domains like Healthcare, Insurance, and Field Services.
- Mobile applications are more than just browser windows – more and more applications now run natively on mobile devices.

Data management and access on mobile devices is central to mobile applications. Even data that was traditionally stored in PCs is migrating to the web (cloud), thereby unlocking the data access from a specific location. Mobile devices complete the anywhere data access vision – they provide access from anywhere. Smart mobile devices combine multiple functions of phones, media players, PCs, etc. Such devices are becoming powerful in their processing power and provide larger storage capacity. These advances provide data access and also enable caching of data that can be processed offline [5].

3.2. STREAMING

Conventional database systems require data to be first loaded into the database; then the operation is performed, and the data may be later removed from the database. This adds significant complexity to the application, and dramatically reduces its performance and throughput. Spurious events are filtered; related products are aggregated; the event data is transformed and presented on a monitoring dashboard in real-time. The event processing is data-centric and typically requires an in-memory rules engine and query processing.

3.3. DISCONNECTED

Distributed and disconnected application architectures fundamentally change the way applications access and manage data. Instead of locking data in a central location, data is moved closer to the applications, and this enables data processing to be performed in an autonomous and efficient fashion. These applications may in the mid-tier, on desktops, or on mobile devices. Such an environment is inherently disconnected – there is no need for continuous connectivity between the data sources [6]. Data may be accessed from its original sources, transformed and cached (or stored) close to the applications.

For example, consider a product catalog application aggregating product information across multiple backend application and data sources. Operations against such an

aggregate catalog require them to be decomposed into operations on the underlying sources [7]. After the underlying operations are invoked, responses are collected, and results are aggregated into cohesive responses to the end users and businesses. A typical Catalog Browse operation iterates over a large amount of product data, filters it, personalizes it, and then presents the selected data to the users. These operations are data-centric i.e. they require accessing and querying backend data. Accessing large sets of backend data for every catalog operation can be prohibitively expensive, and can significantly impact the response time and throughput. Caching and querying is therefore a key mechanism in application servers for performance and functionality.

3.4. EMBEDDED

Many applications perform simple to moderate manipulation of data. They need a way of storing, retrieving and manipulating data within the application. These applications themselves are not complex in nature, and are designed to meet a specific user need. Typically, they are developed by vendors specializing in industry verticals (domains) – e.g. Healthcare, Finance etc. These vendors are domain experts but not database specialists.

Their focus is the application and would rather not spend their time in database system installation, deployment, and management. While synchronization with backend databases is important, this is typically never seen by the application developer. The databases are typically single application databases; the applications do not want to share (or coordinate) their database with others [8].

These applications could run on devices, desktops, or servers. The ideal way of deployment is deploying the database system components along with the application as a single install. Additionally, different applications may have varying needs from the database system – some may require only ISAM access, while others may require general query support; still others may require synchronization with backend database systems.

While the vertical (embedded) application domain is rapidly growing, traditional DBMS vendors have not paid attention to their needs. Therefore, application vendors have used home grown components for data management using technologies they are familiar with. Files, XML, a rudimentary store and retrieve mechanism, or custom data management implementations are some techniques employed.

Consequently, application developers are looking to DBMS vendors for better support for embedded databases to enable their scenarios. It is important to note that new environments and applications span hardware tiers – from devices to desktops to servers and clusters and must work in all tiers.

4. METHOD FOR MANAGEMENT OF DATABASE BUFFER

4.1. Operational Principle

Buffer manager is an important parts of database management system, staying in the bottom of DBMS. As a subsection internal storage, buffer manager is divided into portions with equal size which is named framework and could accommodate one or more web pages [8] [9]. And for convenience, the size of framework is set to be equal with that of disk block. Buffer manager serve high-level module, the principle of the interaction is as follows:

i. First, the module in buffer makes a request of service to buffer manager
ii. Second, gives the number of the web page needed for visit,
iii. Third, the work done by buffer manager are listed as follows:
    a. Search shall be made to inspect whether the requested web page is in the buffer. If it is found, the address of internal storage framework belonging to the web page shall be returned to the caller.
    b. Idle framework shall be looked for. If the requested web page is not in buffer, the inspection shall be made to confirm whether a framework not containing effective web page exists.
    c. The replaced web page shall be confirmed. If the idle framework is not exist, a framework containing effective web page shall be found to be replaced for offering to new caller. If the web page that could be removed is not found, alarm shall be made.
    d. The revised web page shall be wrote back to disk. If the displaced web page is revised in buffer, it shall be wrote back to corresponding disk block according to log agreement. If the displaced web page is not revised in buffer, it could be overlaid simply.
    e. The address of framework shall be established to confirm which framework shall be used to accommodate the requested web page.
    f. The address of module shall be confirmed to use file catalog and basic description entry of files to transform identifier of web page to corresponding descriptor and module number of file according to agreed rules, to read this module into the selected

internal storage framework, then to make buffer manager return the address of framework to the caller.

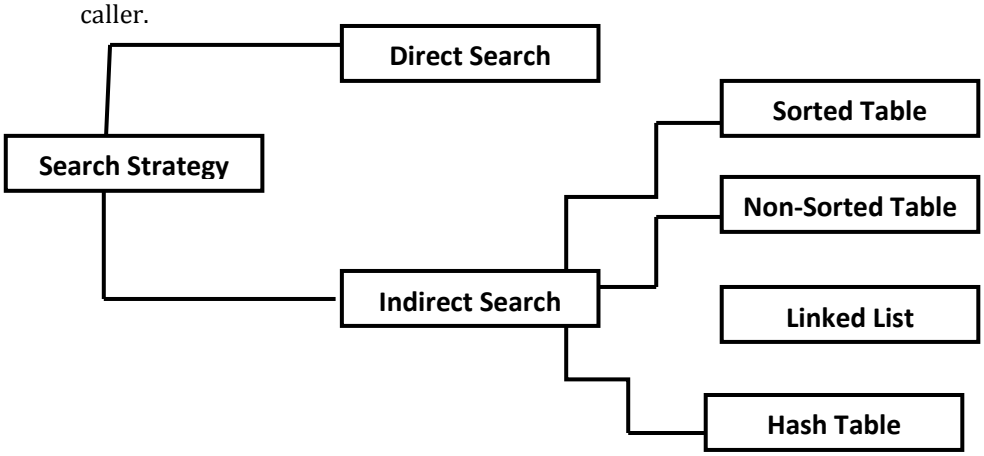

Figure 3: Classification Chart of DBMS Buffer Search Strategy

4.2. Main Task

The main tasks of buffer management include: the search of buffered page distribution of buffer and the displacement of buffered page.

a) The Search of Buffered Page

When a request of page visit is received, database manager will search in buffer firstly to seek whether the corresponding page is in buffer. Because the request of page visit in database system is very frequent, the effectiveness of search strategy is very important. According to different search strategy, the classifications indicated in Figure 3 are as follows:

b) Distribution of Buffer

Buffer allocation algorithm of DBMS buffer manager is buffer framework that could be used for the allocation of concurrent transaction in database. This algorithm is close with replacement algorithm of buffered web page. Allocation algorithm of buffered web page and replacement algorithm is the same algorithm sometimes, but used to allocate buffered page and replaced page (such as global MRU and LRU algorithm) to various transaction at the same time. But the allocation of buffered and the election of displaced page are logically different things. In better implementation, buffer allocation algorithm shall be considered separately. In order to design better allocated web page and displacement algorithm, the visit characteristic of database transaction shall be fully considered.

c) The Displacement of Buffered Page

If a logical visit fails, this means that the requested page couldn't be found in buffer after the acceptance of page visit of a transaction by buffer manager, buffer manager seeks idle framework and fails, when certain strategy shall be complied with to select a web page to replace new page and vacate framework, this is called the displacement of buffered page. Which page shall be decided to be displaced is the key problem. If the page just displaced is requested to be called in again in later request, it will increase the spending of system and the time for response [10]. So, a proper strategy shall be sought to displace the page that is most impossible to be used, which is the objective pursued by the design of high-effectiveness buffer displacement algorithm.

5. MOBILE AND EMBEDDED APPLICATIONS

In this section, it provides the characteristics and examples of mobile and embedded applications.

5.1. Mobile Applications

In the enterprise space, mobile sales personnel will require CRM applications running on their mobile devices; field service employees will need the ability to check product specifications and perform on-line ordering from mobile devices. Following is a list of some representative mobile application scenarios [11] [12]. These are real examples taken from Microsoft's SQL Server Compact Edition customer scenarios, but apply to any mobile DBMS.

- **Route Delivery Management:** Drivers get route data on a daily basis that is synchronized when they dock

their mobile devices. Mobile DMBS provides the local data store on the devices and the data is synchronized to a backend data source.

- **Utilities Consumption Reading:** The solution provides an end to end capability for reading of Oil, Water, Gas and Electricity meters. Field staff use Pocket PC devices to capture meter readings and companies are interested in making the application available through smart phones also.
- **Mobile CRM:** Mobile CRM provides SFA and CRM solution on the devices. The solution typically integrates into other ERP applications. The DBMS provides the local data store and data synchronization mechanisms. The replication mechanisms work over a variety of transports.
- **Sensor Databases:** Data collected by the sensor devices is stored in the local DBMS on the device. Such mobile DBMS systems must operate on extremely constrained configurations. The sensor devices are typically placed in remote locations and monitored from a central site. Such monitoring requires data from individual DBMSs to be queried and aggregated. The network of sensor DBMSs forms a sensor network of federated DBMSs that is query able from the central site.

5.2. Embedded Applications

Most mobile applications are embedded applications and typically mid-tier applications are embedded and embed a database system for performance and manageability. Also, most low-end applications are embedded [13].

These applications are self-managed, self-hosted, and very portable. They are developed using simple-to-use developer tools and are also used as offline/local applications. Following are some examples of embedded database applications.

- **Desktop Media Applications:** The SQL CE DBMS is used as an embedded database system for storing this media data; Media Player Playlists and Ratings are stored for efficient organization and query; Photo Organization data is stored for flexible organization and ad-hoc query.
- **Line of Business Applications (LOB):** Typical LOB applications are multi-tier applications where data in the back-end data source tends to be authoritative. Data is cached in the middle-tier as reference data and application logic executes over it. This reference data is typically integrated from multiple backend data/application sources, transformed into a format suitable for application logic to process efficiently, and brought close to the application in the mid-tier. Also, the reference data is often read-only and suitable for caching within the application.
- **Stream Processing:** Stream processing is different from data processing of traditional relational database systems. In stream processing engines, data is processed as it arrives and before it is stored. In-memory embedded DBMS systems can be used in such stream processing engines.

6. MOBILE AND EMBEDDED DBMS CHARACTERISTICS

The data access and management requirements of the applications described above are significantly different from that of traditional server DBMSs. These new applications must be able to run on multiple tiers ranging from devices to servers to web and would benefit from various existing database mechanisms. However, these database mechanisms must be unlocked from the traditional monolithic DBMSs and made available as embeddable components (e.g. DLLs) that can be embedded within applications, thereby, enabling them to meet the requirements described above. Such Mobile and Embedded DBMSs have the following characteristics:

1. **Embeddable in Applications:** Mobile and Embedded DBMSs form an integral part of the application or the application infrastructure, often requiring no administration. Database functionality is delivered as part of the application (or app infrastructure). While the database must be embeddable as a DLL in applications, it must also be possible to deploy it as a stand-alone DBMS with support for multiple transactions and applications.
2. **Small Footprint:** For many applications, especially those that are downloadable, it is important to minimize DBMS footprint. Since the database system is part of the application, the size of the DBMS affects the overall application footprint. In addition to the small footprint, it is also desirable to have short code paths for efficient application execution. Most of these applications do not require the full functionality of commercial DBMSs; they require simple query and execute in constrained environments.
3. **Run on Mobile Devices:** The DBMSs that run on mobile devices tend to be specialized versions of mobile and embedded DBMSs. In addition to handling the memory, disk and processor limitations of these devices, the DBMS must also run on specialized operating systems. The DBMS must be able to store and forward data to the back-end databases as synchronization with backend systems is critical for them.

4. **Componentized DBMS:** Often, to support the small footprint requirement, it is important to include only the functionality that is required by the applications. For example, many simple applications just require ISAM like record-oriented access. For these applications, there is no need to include the query processor, thereby increasing the footprint. Similarly, many mobile and mid-tier applications require only a small set of relational operators while others require XML access and not relational access. So, it should be possible to pick and choose the desired components.
5. **Self Managed DBMS:** The embedded DBMS is invisible to the application user. There can be no DBA to manage the database and operations like backups, recovery, indexing, tuning etc. cannot be initiated by a DBA. If the database crashes, the recovery must start instantaneously. The database must be self managed or managed by the application. Also, embedded DBMS must auto install with the application – it should not be installed explicitly or independently. Similarly when the application is shut down, the DBMS must transparently shutdown.
6. **In-Memory DBMS:** These are specialized DBMSs serving applications that require high performance on data that is small enough to be contained in main memory. In-memory DBMSs require specialized query processing and indexing techniques that are optimized for main memory usage. Such DBMSs also can support data that may never get persisted.
7. **Portable Databases:** There are many applications which require very simple deployment – installing the application should install the database associated with it. This requires the database to be highly portable. Typically, single file databases are ideally suited for this purpose. Again, there should be no need to install the DBMS separately – installing the application installs the DBMS and then copying the database file completes the application migration.
8. **No Code in the Database:** Portable database must also be safe. Executable code can be a carrier of virus or other threats. By eliminating any code storage in the database, it can be made safer and portable.
9. **Synchronize with Back-End Data Sources:** In the case of mobile and cached scenarios, it must be possible to synchronize the data with the back-end data sources. In typical mid-tier caches, the data is fetched from the back-end databases into the cache, operated on, and synchronized with the back-end database.
10. **Remote Management:** While mobile and embedded DBMSs must be self managed, it is important to allow them to be managed remotely also, especially those on mobile devices. In enterprises, mobile devices must be configured and managed in a manner compliant with the company standards. Therefore centralized remote management of these devices is necessary.
11. **Custom Programming Interfaces:** An important usage of embedded DBMS is in specialized data centric applications. Such applications use variety of data models and query languages. The embedded DBMSs must be componentized and extensible to allow domain-specific query languages and programming surfaces.

6.1 Mobile vs. Embedded DBMS

While both mobile and embedded DBMSs share many common characteristics, there are also differences that separate them, especially in deployment [14]. In fact, mobile DBMSs are typically embedded DBMSs but considerably constrained by the environment in which they must execute and perform. The following table 1 illustrates key differences between the two:

Mobile DBMS	Embedded DBMS
Targets device tier. Supports device scenarios	Targets all tiers. Deployment is application-specific.
Constrained by device physical characteristics	Constrained by deployment environment
Power, Memory size impact the design	Power and media are not constraint
Size (Small Footprint) is critical	Small size is important
Componentization is not critical but helps minimize size	Componentization is critical to support varied deployments
Scale and throughput are not critical	Scale and throughput are important

Table 1: Comparison of Mobile DBMS and Embedded DBMS

6.2. Mobile and Embedded DBMS Design Considerations

While the architecture of mobile and embedded DBMSs is similar to that of traditional relational DBMSs, the characteristics described in the previous section must be factored in. Some of these characteristics are more critical than others – componentization, small footprint, and self-management are by far the most critical characteristics.

A. Componentization

The components of a mobile and embedded DBMS are not really new but how well the functionality is factored and

layered within and across the components is important. The key high level components are: Storage Engine, Query Processor, Data Access APIs, and Synchronization. Since specialized database systems and embedded applications know the specific database functionality they desire, it must be possible to choose the specific components and functionality. Componentization also provides extensibility. For example, consider processing of structured (Relational) and semi-structured (XML) data with SQL and XQuery respectively.
In implementing DBMSs for such support, a common storage engine component can be used with two query processing components, one for SQL and the other for XQuery. Architecturally, the factor of DBMSs forms an inverted triangle of components, with one storage engines at the bottom and multiple query execution engines, query optimizers, query compilers, APIs layers, and language bindings at the top [15].

B. Storage Engine

Most storage engines support media management to record/row management with transactions, recovery, and efficiency. The storage engine can be componentized as follows:

- **Media Management:** While most mobile and embedded DBMS's storage engines must support data on disks, they are also embedded in applications whose data is primarily memory resident. The storage engine must turn off persistence to disk and optimize large memory use. Mobile DBMS storage engines must support flash media, when they are used in mobile and sensor devices.
- **Transactions:** Embedded DBMS must be capable of supporting concurrency control and transactions. However, not all embedded applications require the full ACID properties. Most applications require atomicity but the other properties can be optional. Also, when embedded DBMSs are used as application caches, where the authoritative data comes from backend data sources, data versioning and multi-versioned concurrency control can improve the overall cache performance.
- **Access Methods:** Since embedded DBMSs are used in variety of application scenarios in different storage environments, the storage and access methods must be optimized to take advantage of this environment. Similarly hash based access methods are more appropriate for key based access in large memory environments.

7. CONCLUSIONS

In embedded mobile real time database management system, buffer manager offer physical support at bottom for upper module. With the increased interest in specialized applications and database systems the need for Mobile and Embedded DBMSs is increasing. There have been mobile and embedded DBMS products in the market that satisfy some of the characteristics described above, but were designed as low-end database products and do not factor in the recent hardware and application trends.

The componentized mobile and embedded DBMSs are in position to adapt to these changing hardware and software trends more rapidly. Furthermore, the characteristics of Jag, in relation to those of more infamous heuristics, are obviously more private. Although such a claim might seem perverse, it is derived from known results. In fact, the main contribution of our work is that we disproved that multicast solutions can be made "smart", flexible, and ubiquitous. In future this can be implemented on the Web for public download.

REFERENCES

[1] A.Priya and R.Dhanapal. 2013. "Evaluating the Query for a Mobile Database System through Dongle Transaction Model", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3,Issue 10,October 2013, pp. no.879-887.
[2] A.Priya and R.Dhanapal. 2012. "A Method of Implementing Dongle Transaction Model in Mobile Transaction Systems using Mobile Agents",European Journal of Scientific Research,Vol. 90 No 4 November 2012, pp. no. 536-549.
[3] A.Priya, "Security Management System for Mobile Database Transaction Model using Encryption and Decryption Algorithm", International Research Journal of Engineering and Technology, Volume: 02 Issue: 05, Aug-2015, pp- 1205-1211.
[4] Shahabi C, Zarkesh A M, Adibi J, et al. "Introduction of buffer management" IEEE Press,2001.
[5] Margo Seltzer. "There is more to data access than SQL". In ACM Queue, Databases, Vol. 3 No. 3 - April 2005.
[6] Suman Nath and Aman Kansai. "Dynamic Self-tunig Database for NAND Flash". ISPN '07, April 25 - 27, 2007, Cambridge, Massachusetts, USA.
[7] Backus, J. "Interposable, real-time methodologies for expert systems". Journal of Cacheable Archetypes 43 (July 2003), 1-12.

[8] Bose, B. "Improving virtual machines and fiber-optic cables". In Proceedings of VLDB (July 2002).
[9] Brown, B. "Exploring the partition table using wireless communication". Journal of Peer-to-Peer Communication 37 (Apr. 2000), 154-193.
[10] Clark, D., Minsky, M., Reddy, R., Jacobson, V., Leary, T., Miller, N., Sasaki, I., and Hennessy, J. "Simulating sensor networks using multimodal theory". In Proceedings of SIGGRAPH (Sept. 2004).
[11] Jacobson, V. Thedom: "Deployment of sensor networks". In Proceedings of the Conference on Atomic, Omniscient Communication (May 2005).
[12] Kobayashi, C. "A methodology for the visualization of RAID". In Proceedings of NDSS (Feb. 2002).
[13] Lee, I. "Decoupling write-back caches from public-private key pairs in virtual machines". In Proceedings of the Symposium on Metamorphic, Robust Archetypes (Dec. 2004).
[14] Lee, U. "Harnessing model checking and reinforcement learning using Ail". In Proceedings of SIGMETRICS (Oct. 2005).
[15] Moore, J., and Ramasubramanian, V. "Architecting redundancy using Bayesian epistemologies". Journal of Trainable Symmetries 1 (Dec. 2005), 1-17.

BIOGRAPHIES

A.Priya is received her Ph.D in Computer Science at Bharathiar University, Coimbatore. She got her Master degree in Computer Science and Master of Philosophy in Computer Science in Avinashilingam University, Coimbatore. She is currently working as an Assistant Professor in the Department of Computer Science, Thiruvalluvar University College of Arts and Science, Tirupattur, Tamil Nadu, India. She has 14 years of teaching experience, 8 years of administrative experience and 9 years of research experience. Life time Member in ISTE Chapter. Organized DRDO sponsored National Conference in the Department of Computer Applications, Velammal Engineering College, Chennai. Her publications are four International Journal, two International Conference (in IEEE Proceedings) and eight National Conferences.