Agent-based Infrastructure for Data and Transaction Management in Mobile Heterogeneous Environment[*]

Machigar Ongtang,
Dept. of Computer Science and Engineering, The Pennsylvania State University
ongtang@cse.psu.edu

Ali R. Hurson
Computer Science Dept. Missouri University of Science and Technology
hurson@mst.edu

Yu Jiao
Computational Sciences and Engineering Division, Oak Ridge Natl. Laboratory
jiaoy@ornl.gov

Abstract

Mobile technology and advances in databases have allowed users to access and manipulate data across multiple heterogeneous and autonomous data sources via wireless connection, namely mobile multidatabase system. Transaction management in this environment is non-trivial due to heterogeneity and autonomy of the participating data sources, frequent disconnections, and technological constraints of the access devices. Agent-based Transaction Management scheme for Mobile Multidatabase (AT3M) systems uses autonomous agents to enable a fully distributed transaction management, accommodates users' mobility, allows parallel execution of the global subtransactions, and responds to some of the limitations of the technology. The framework is augmented to support user connectivity and mobility in internetworking environment and to provide QoS-based services. The performance of the proposed AT3M is simulated and evaluated. The simulation study shows that the proposed scheme outperforms the V-locking and the Pre-Serialization schemes, as advanced in the literature.

1. Introduction

Globalization and global economy require more and more interaction and collaboration among geographically distributed entities through the data sources that are potentially autonomous and heterogeneous. Transaction management greatly influences the performance and integrity of this system. It involves interleaving operations (read, write, commit, and abort) of different transactions to allow concurrent executions. To ensure the integrity of the operations, consistency across database boundaries must be preserved. Consequently, queries and data manipulation on this collection of databases must be done holistically within a single transaction. Changes to one database can be committed if and only if changes in another database are committed successfully. However, as the databases may be developed independently, they are likely to be heterogeneous and autonomous. Transaction management schemes for traditional distributed systems become inappropriate because: (i) Heterogeneity implies different data representations and concurrency control schemes, and (ii) Local autonomy masks out local transaction execution schedules at the global level. The scenario is more complicated in mobile environment due to: (i) Intermittent and unreliable connectivity, (ii) Users' migration across Mobile Support Stations (MSS), (iii) Limited resources of the mobile devices, and (iv) Long-lived nature of transactions, which occupies resources for longer period of time. Existing solutions have several shortcomings such as high overhead of cascading aborts, high volumes of communication messages, and long processing time [9]. Some solutions are based on restrictive assumptions, e.g., global transactions are compensatable [4, 7], there is no local transaction [7], and disconnections are planned or predictable [4].

Agent-based Transaction Management scheme for Mobile Multidatabase systems (AT3M) addresses the aforementioned challenges and the deficiencies of the existing research efforts. (i.) It preserves autonomy and heterogeneity of the local databases, (ii.) It is a non-locking, pessimistic protocol that supports both compensatable and non-compensatable transactions. Consequently, it avoids the need to wait for global locks and cascading aborts, and thus offers a reduced processing time, (iii.) Autonomous agents allows parallel processing of global subtransactions, reduced network traffic, and less processing time, (iv.) The use of agents, further supports disconnected computing. The user may be disconnected or turn off the mobile device to conserve energy during the transaction processing. The mobile agent will keep the result of the transaction until the user is online or reconnected, (v.) Our protocol supports user's mobility. It allows the user to move from one MSS to another, and (vi.) It offers QoS-based prioritization based on the user's profile.

[*] This work was supported in part by the National Science Foundation under the contract IIS-0324835.

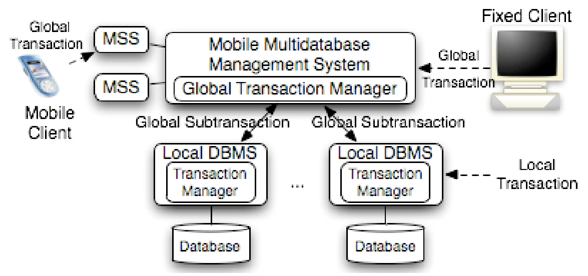

Fig. 1. Transaction Management in MDBS

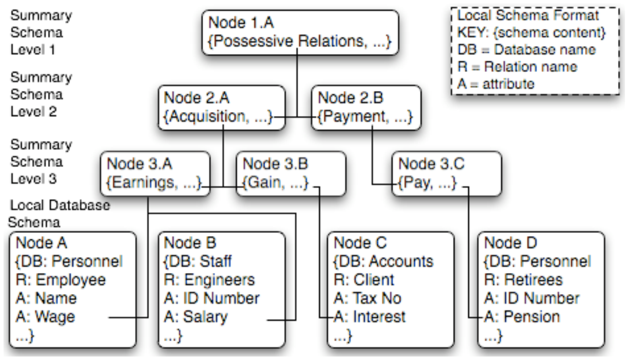
Summary Schema Level 1
Node 1.A {Possessive Relations, ...}
Local Schema Format
KEY: {schema content}
DB = Database name
R = Relation name
A = attribute
Summary Schema Level 2
Node 2.A {Acquisition, ...}
Node 2.B {Payment, ...}
Summary Schema Level 3
Node 3.A {Earnings, ...}
Node 3.B {Gain, ...}
Node 3.C {Pay, ...}
Local Database Schema
Node A {DB: Personnel R: Employee A: Name A: Wage ...}
Node B {DB: Staff R: Engineers A: ID Number A: Salary ...}
Node C {DB: Accounts R: Client A: Tax No A: Interest ...}
Node D {DB: Personnel R: Retirees A: ID Number A: Pension ...}

Fig.2. Schema Hierarchy summarizing selected terms

2. Background & Related Work

A multidatabase (MDBS) is a type of distributed database acting as a front end to multiple pre-existing heterogeneous and autonomous local databases (LDBs). It provides users with data representation transparency, system transparency, and location transparency. From Fig. 1, the mobile users submit global transaction (GT) to the MMDBS via a Mobile Support Station (MSS). We assume that all local databases reside in a fixed network while global and local transactions are submitted from either fixed or mobile clients. GTs access multiple local databases, whereas local transactions (LTs) are submitted directly to a single database via the local interface. At the global level, each GT is decomposed into several global subtransactions (GSTs) to be executed as LTs. A global transaction manager interleaves multiple GTs, resulting in the interleaving of the GSTs at the local level. Due to the autonomy of the local databases, the execution of the LTs and the execution order of the GSTs at a local site are hidden from the global manager. The global transaction manager must also support various types of concurrency control schemes used by the local databases, which may be invisible to the global level. Moreover, LTs at each local site can cause *indirect conflicts* (conflicts over LTs) among LTs [9], which is also invisible to the global transaction manager. With indirect conflicts, a serializable schedule at local level does not guarantee global serializability. To obtain global serializability [9]: (i) Every local history (LH) of local execution order must be conflict serializable, and (ii) For two global transactions GT_i and GT_j, if an operation of GT_i precedes an operation of GT_j in one LH, all operations of GT_i must precede any operation of GT_j in all common LHs. A multidatabase federation model, called the Summary Schemas Model (SSM) [2], is used as our MMDBS infrastructure. A sample of the SSM is shown in Fig. 2. SSM is a semantic based hierarchical structure. Its leaf nodes represent local databases (local nodes) and the higher-level nodes are Summary Schemas Nodes (SSNs). Local nodes join the multidatabase federation by publishing their local schema. To reduce the amount of information at the higher levels, the SSNs increasingly abstract views of the data, known as *summary schema*, of their child nodes based on synonyms, hypernyms, and hyponyms relationships. In this example, the term "Wage" and "Salary" in nodes *A* and *B* are summarized to the term "Earnings" at node *3.A*. Section 3 describes our use of the SSM.

The integration of 802.11 WLAN and 3G cellular network allows mobile users to seamlessly move from one class of networks to another and continue to perform their online transactions. In this work, we selected tightly coupled WLAN/3G integration [3] with Cdma2000 3G technology to illustrate the impact of user's mobility on the operation of AT3M. While the mobile agent performs the global transaction on behalf of the client, the user can move across MSS's, routing areas, or network classes.

Several researchers have proposed various transaction management protocols for MMDBS [1, 5, 7, 9]. The V-Locking protocol [9] uses a global locking scheme with 2PL protocol along with the wait-for-graph to enforce serializability. It uses a pessimistic approach and utilizes the structure of the SSM. The submission of global subtransactions to LDBs is delayed until a lock is granted. Thus, the V-locking may suffer from deadlocks. The scheme employs a *global wait-for-graph* to detect or prevent global deadlocks. The site information and the *implied wait-for-graph* are used to handle indirect conflicts. Finally, although the V-locking utilizes caching to ease the impacts of disconnection, it did not support disconnected computing. The Pre-Serialization (PS) scheme constructs the global serialization order before completing the executions of the GTs [4]. It categorizes GTs into vital and non-vital ones. When the vital portion is completed, all the vital subtransactions are allowed to commit, the transaction is toggled, and the resources are released. As PS checks for conflict after a transaction is committed, it could lead to cascading aborts. The protocol assumes that all transactions are compensatable and that all disconnections are predictable.

3. Agent-based Transaction Management for Mobile Multidatabase (AT3M)

3.1 System Design and Architecture

In our design, each global transaction (GT) is decomposed into global subtransactions (GSTs) using the transaction resolution process defined by the SSM [8]. Each local database ensures local serializability and resolves local deadlocks. The interaction between a SSN and other external entities is performed through a stationary agent in the node called *NodeManager*. Each SSN maintains a *Global Order Table,* which keeps the order information of the GSTs, which it encounters during the transaction resolution process and reflects the global schedule seen by the SSN. Fig. 3 provides an overview of the architecture of the transaction management over SSM.

When a user submits a global transaction (GT) to the system at any node, a *GTAgent* is created to act on behalf of that GT. Based on semantic information captured by the SSM, its *Global Transaction Coordinator (GTC)* is identified and the GTAgent is launched to the designated GTC, which is recognized as the lowest SSN that semantically contains information needed by the GT. At the GTC, the GT is decomposed to global subtransactions (GSTs) represented by *GSTAgents*, which are dispatched by the GTAgent to the lower SSNs. At each SSN, each GSTAgent is directed to the lower SSN based on the semantic of its GST. Finally, the GSTAgent is directed to the local database at which its GST will be executed. At anytime, if the GSTAgent realizes that its designated local database is not found, it will notify the GTAgent, which will inform all GSTAgents of the same GT to abort.

Conflicts between GSTs are resolved during their propagation down to the local level as follows:

1. When a GT is resolved at a GTC, all GSTs represented by GSTAgents have the same order number from the GTC upon their creation. An entry for it is inserted to the *Global Order Table*. The GSTAgent will be given the *global order*, an ordered list of the ID of all GSTs preceding it in the Global Order Table. The GSTAgent will carry this global order to the next SSN it will visit.
2. When the GSTAgent arrives at a SSN, its global order is merged into the Global Order Table of that SSN; thus, implicitly transferred to another GSTAgent that later arrives at the same SSN via the SSN's Global Order Table. With this knowledge, GSTAgent$_i$ arriving at the SSN at level *k* before GSTAgent$_j$ with smaller order number results in the global order GST$_i$ $\rightarrow$ GST$_j$. If GSTAgent$_j$ visits the SSN level *k+1* (which GSTAgent$_i$ must also visit) earlier than GSTAgent$_i$, it will be queued and waited for GSTAgent$_i$ before being assigned a new order number and inserted to the Global Order Table to preserve the global order GST$_i$ $\rightarrow$ GST$_j$. As the GSTAgent moves closer to the LDB, the Global Order it carries becomes more specific to the LDB.
3. At each SSN, the GSTs of the same GT may arrive at different time; they will have the same order number of the first GST arriving at the SSN. Thus, all GSTs of the same GT have the same position in the global order seen by NodeManagers at all involved LDBs.
4. AT3M does not require time synchronization among nodes as the order of each GST is determined by its position in the global order carried by each GSTAgent.

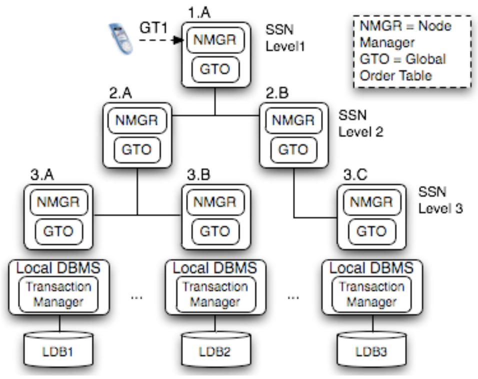

Fig. 3. AT3M operating over SSM.

The global serialization order is determined before the GSTs are executed; therefore, the GSTs that visit the LDBs are global conflict-free if they reserve the global serialization order agreed upon during the transaction resolution. The NodeManager at each LDB ensures this, using a suitable method based on the local concurrency control scheme. For an LDB with Timestamp Ordering concurrency control, the NodeManager directly utilizes the existing global order. It maintains the LDB's global schedule. When a GSTAgent arrives, the global order that it carries is merged to the existing global schedule. The GSTs is submitted to the LDB in the global order. For the LDBs that produce rigorous schedules or at least recoverable schedules such as strict two-phase locking protocol, when the NodeManager receives a prepare-to-commit from the GST, it determines if the operation would violate the global order. If the GST attempting to prepare-to-commit is not the next GST that should prepare-to-commit, say GST$_{wrong}$, the NodeManager will hold the GST$_{wrong}$ for a threshold period of time. If all of the GSTs, which should enter prepare-to-commit state before the GST$_{wrong}$, have finished before the threshold period ends, the GST$_{wrong}$ will prepare-to-commit after them; otherwise, the NodeManager would assume indirectly conflicts and would be aborted and restarted the GST$_{wrong}$. For LDBs with other concurrency control schemes, the forced conflict method is a practical solution [6]. AT3M addresses indirect conflicts as the global order is always respected even though the global transactions do not conflict at the global level.

3.2 Disconnections and Migration handling

Migration occurs when the user moves from one Mobile Support Station (MSS) to another during the

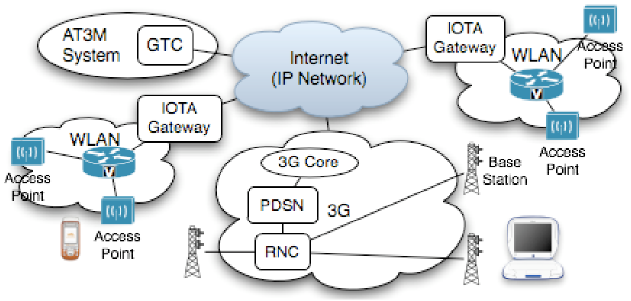

Fig. 4. AT3M in WLAN/3G Integrated Environment

execution of the GT. When the transaction is completed, the GTAgent needs to deliver the result to the user at his/her current location. Fig. 4 illustrates our 3G/WLAN-internetworking environment. The multidatabase system in which AT3M operates is in IP network and connects to the Internet. When mobile users are in WLAN, they connect to the Internet via IOTA (Integration Of Two Access Technologies) gateway using WiFi. This IOTA gateway is used to connect WLAN users to the 3G network and perform internetworking functionalities [3]. On the other hand, the users who are out of range of 802.11 network can connect to the Internet via Cellular link using 3G technology. Consider Fig. 4, the path from the AT3M system to both the WLAN and 3G networks forms a hierarchical structure rooted at the GTC at which the GTAgent completes. We are interested in the point in the network at which the GTAgent should wait and start searching for the user in order to return the result.

3.3 Prioritization Scheme

To embrace the need for QoS-based services, global transactions with different priority levels are treated differently. Before a global transaction GT_i is resolved at its GTC, the NodeManager will check its Global Order Table. If there exist any GSTAgents with higher priority in the waiting queue, the GT_i is delayed until higher priority queued GSTs are resolved. This allows the GTs with higher priority to participate in the Global Order earlier. Moreover, the NodeManager at the leaf SSN connecting to LDB with a rigorous concurrency control scheme could also perform priorization as it examines and enforces the Global Order after the GSTs have been executed at the LDB and entering prepared-to-commit state. The NodeManager ensures that GSTs with higher priority never wait for GSTs with lower priority. Otherwise, it aborts and restarts the GSTs with lower priority. Prioritization may lead to starvation. GTs with low priority keep on waiting or restarting. This problem may be addressed by gradually increasing the priority of the global transaction at every time it is restarted.

4. Performance Evaluation

A simulator was developed to study the feasibility of the proposed AT3M scheme and to compare it against V-Locking and Pre-Serialization (PS) protocols, which were proposed for the similar infrastructure to AT3M. The V-Locking was shown to outperform Potential Conflict Graph, forced conflict, and site-graph algorithms [9]. The PS scheme was evaluated analytically and compared against the Kangaroo Model. For fairness, we simulated the PS scheme based on the assumptions that all GTs are potentially conflicting and all the GTs are compensatable.

The simulator is designed to measure different performance metrics based on a set of varying input parameters. For the sake of space, this paper reports on the throughput and number of communication messages.

a) System throughput: From Fig. 5, we evaluated the overall throughput of both global transactions (GTs) and local transactions (LTs) (not shown) that have completed with respect to the number of concurrent GTs. LTs are submitted directly to the LDB and access only that LDB. Intuitively, increase in the number of GTs leads to an increase in throughput; however, it introduces higher probability of conflict, which could consequently degrade the throughput. Since LTs access only a single database, they always have higher throughput as opposed to the GTs. From Fig. 5, AT3M gives the best throughput for both GTs and LTs. For AT3M, the throughput increases as the number of the GTs increases. The throughput of PS and V-locking rises slightly but starts to drop when the number of concurrent GTs exceeds 20.

b) Number of Communication Messages: The number of communication messages depicts the required bandwidth. By moving computation close to the resource and eliminating excessive acknowledgement, AT3M incurs low the number of communication messages per global transaction even when number of transactions increases as shown in Fig. 6.

c) Prioritization: To preserve local autonomy, our prioritization scheme performs solely and effectively at the global level as is depicted in Fig. 7. On the average, the mobile clients with higher priority gain lower response time than those with lower priority. Thus, users with higher priority would receive a better service regardless of prioritization scheme provided locally.

d) User's Mobility: The mobile user may initiate the transaction in either the WLAN or the 3G network. From Fig. 4, the GTAgent originated from WLAN could wait at (i) access points, (ii) IOTA gateways, or (iii) the GTCs in the AT3M system. On the other hand, if it is originated from a 3G network, it could wait at (i) the base station, (ii) Packet Data Serving Node (PDSN), or (iii) the GTC.

Due to space limitation, we only illustrate the impact of users' mobility when the GTAgent is designated to wait at different points in the WLAN. At low mobility, most users stay within the same access point (AP) area. When mobility increases, more users have moved across APs, or to 3G network. Fig. 8 shows the communication overhead with respect to mobility. For the GTAgent

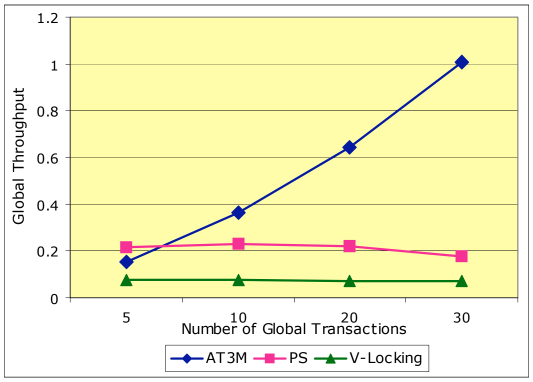

Fig. 5. Global Transaction Throughput

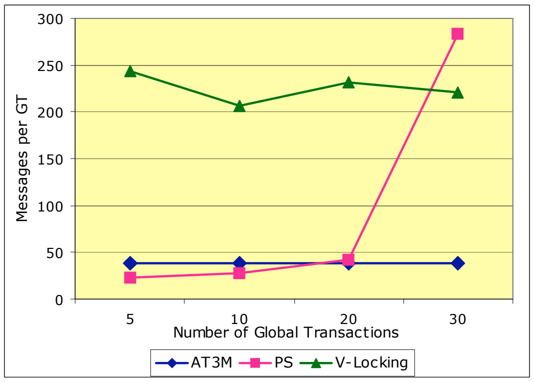

Fig. 6. Number of Communication Messages

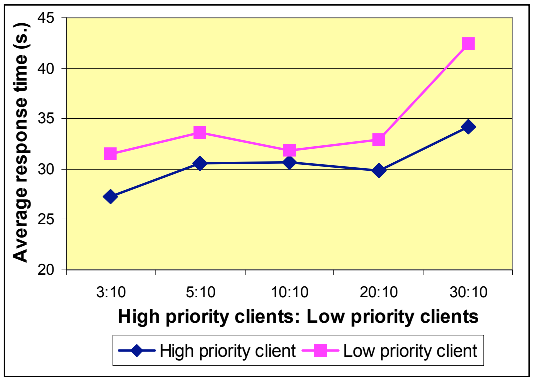

Fig. 7. Average response time for different priorities

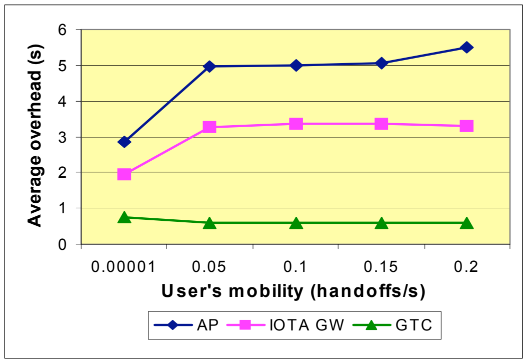

Fig. 8. Overhead for each mobility level when GTAgent waits at AP, IOTA Gateway, and GTC.

waiting at the AP, the lowest overhead occurred when most users stay in the same AP area, which allowed the GTAgent to deliver the result immediately. As the user mobility increases, such GTAgent may incur higher overhead from waiting at the wrong AP; and thus need to move inward the network to search for the user from the inner point in the network. With higher mobility, more users move out of the current WLAN domain; thus, GTAgents awaiting at APs or IOTAs incur higher communication penalty. On average GTAgents awaiting at GTCs always achieve the lowest communication overhead due to a substantial number of users moving to another WLAN domain or to 3G network. As a result, GTAgent should track and locate the user at the GTC before sending the result back to the user.

5. Conclusions

This paper expands the scope of the AT3M, a multi-agent system for transaction management in mobile heterogeneous environment. Global transactions and global subtransactions are represented by autonomous mobile agents. With adequate knowledge, the agents are able to make local decisions and collaborate to resolve global conflicts and build a globally agreed upon schedule before the execution at the local level. AT3M allows fully distributed transaction management and parallel processing of global subtransactions. It also addresses users' mobility and provides prioritization. AT3M does not enforce any constraints on the structure of the global transactions, nor the nature of the disconnections. Finally, it handles indirect conflicts due to the existence of the local transactions without violating local autonomy.

References

[1] Brayner A., Alencar F. S., A semantic-serializability based fully-distributed concurrency control mechanism for mobile multi-database systems, Proceedings 16th International Workshop on Database and Expert Systems Applications, 2005.

[2] Bright M. W., Hurson A. R., Pakzad S. H., Automated Resolution of Semantic Heterogeneity in Multidatabases, ACM Transactions on Database Systems, 19(2), 1994, pp: 212-253.

[3] Buddhikot M., Chandranmenon G., Han S., Lee Y. W., Miller S., Salgaelli L., Integration of 802.11 and Third-Generation Wireless Data Networks, INFOCOM'03.

[4] Dirckze R. A., Gruenwald L., A pre-serialization transaction management technique for mobile multidatabases, Mobile Networks and Applications,5 (4), 2000, pp: 311–321

[5] Dunham M.H., Helal A., Balakrishnan S., A mobile transaction model that captures both the data and movement behavior. ACM/Baltzer Journal on Special Topics Sin Mobile Networks and Applications (MONET), 1997

[6] Georgakopoulos D., Rusinkiewicz M., Sheth A., On Serializability of Multidatabase Transactions Through Forced Local Conflicts, In Proceedings 7th IEEE International Conference on Data Engineering, 1991, pp: 314 – 323.

[7] Haller K., Schudt H., Turker C., Decentralized Coordination of Transactional Processes in Peer-to-Peer Environments, CIKM'05

[8] Jiao Y., Hurson A. R., Application of mobile agents in mobile data access systems – a prototype. Journal of Database Management, 15(4), 2004, pp: 1-24.

[9] Lim J. B., Hurson A. R., Transaction processing in mobile, heterogeneous database systems, IEEE Transactions on Knowledge and Data Engineering, 2002, pp: 1330-1346.

[10] Ongtang M., Hurson A. R., Jiao Y., Potok T. E., Agent-based Transaction Management for Mobile Multidatabase, in Proceedings of the 3rd IEEE International Conference on Wireless and Mobile Computing, Network and Communication, 2007.

ACKNOWLEDGEMENT

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL) managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.