DAMAGE DISCOVERY IN DISTRIBUTED DATABASE SYSTEMS

Yanjun Zuo and Brajendra Panda

Abstract Damage assessment and recovery in a distributed database system in a post information attack detection scenario is a complicated process due to the indirect dependencies among (sub) transactions that are executed in various sites. Particularly, damage assessment in such a system requires collaborations among multiple participant sites as a result of distributed transactions. In this paper, we discuss two primary models, namely, centralized and peer-to-peer, to conduct damage assessment after an intrusion on a distributed database system is reported. For the centralized model, three different options have been presented. Advantages and disadvantages of each model are discussed.

1. INTRODUCTION

Distributed database systems are widely used in web based applications and mission-critical programs. Reliability and higher availability of resources are major motivations for moving toward distributed database management systems. More importantly, the introduction of a wide range of real-time applications can be implemented more effectively in a distributed environment. However, distributed database systems introduce difficulties in data protection and overall system securities. In the case of detection of an intrusion in such a system, immediate damage assessment and recovery must be followed in order to completely wipeout the effect of the attack and restore the database to a consistent state, the state the database would have reached if there were no attack. Assessment is important since initial damage could later spread to other parts of the database via legitimate transactions or other means such as system integrity check as described in [1] and [3]. During the assessment and recovery process the system must be offline or at least part of the system will not be available for the users. This affects the system availability and introduces the known problem of denial

of service. In distributed database systems, due to heavy dependencies among transactions and sub-transactions at various sites, damages would spread faster than that in a centralized system. Therefore, it is essential to employ fast and accurate damage assessment procedures as soon as the attack is detected.

2. RELATED WORK

Several models have been proposed for database damage assessment and recovery. Some of the recent developments on damage assessment and recovery from information attacks are discussed in [4, 8, 9, 10, 11]. Most of those proposed models concentrate on centralized database systems and very few of them have been applied to distributed database systems. Assessing damage for a centralized database system is relatively easier since there is no distributed transaction, which may introduce "hidden" damage that could not be detected if viewed only from any single site after an attack has been detected. Peng Liu and Xu Hao [7] have proposed a completely distributed algorithm for distributed database systems. Their model essentially requires communications among every site directly. In a system with a large number of distributed sites, the model incurs a large volume of network communications in order to detect all affected transactions. In this work, we have developed damage assessment models, which fall into two basic categories: centralized and peer-to-peer. The former model can effectively reduce the network traffics required for communications among multiple sites for the purpose of damage assessment and the latter avoids single point of failure and offers faster execution.

3. BASIC DAMAGE ASSESSMENT MODELS

Our developed models are based on the assumptions that the local logs are not damaged and blind writes are not permitted. The output of each model is a set of transactions that are detected as affected, either directly or indirectly, by a malicious transaction. These transactions are then provided as input to a recovery algorithm. Recovery methods are not discussed in this paper due to space constraints. We assume that the distributed database system consists of multiple sites, each of which contains a local manager to coordinate with other sites and/or the coordinator. The two basic assessment models we present are centralized and peer-to-peer models as illustrated in Figure 1. We have developed algorithms for each of the models as well as their sub models (where applicable). Due to space limitations, the

algorithms are not provided in this paper; interested readers may contact the authors.

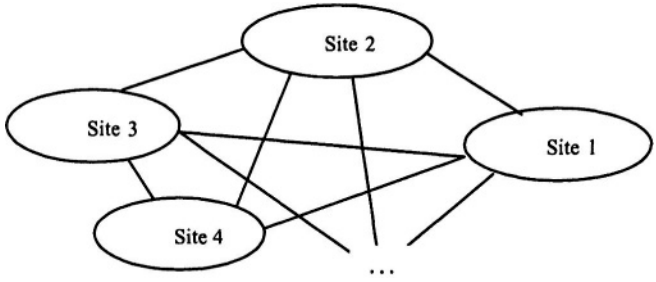

Figure 1(a): Peer-to-peer model

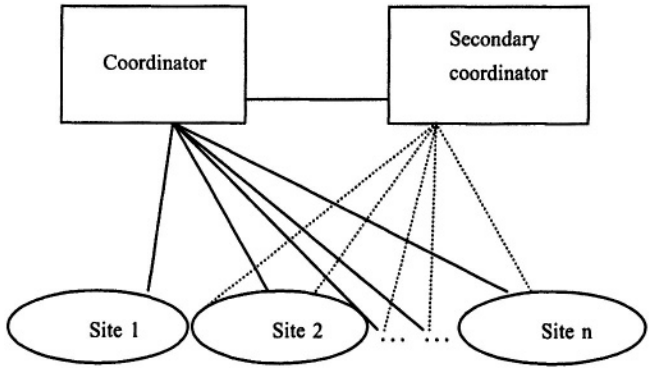

Figure 1(b): Centralized model

3.1 Peer-to-peer model

This model does not require a coordinator in order to perform damage assessment. Every site acts as a peer to others. Each site manager is responsible for scanning its local log. If the site manager identifies some affected (global) transactions, it multicasts their identifiers to every other site, where a sub-transaction of any of these affected global transactions is executed. We assume that each site keeps information about sites where a global transaction's sub-transactions were executed. Then each receiving site manager scans its local log to further detect any more affected transactions based on the newly received information if there are any. Each site manager starts scanning the log beginning where the first affected

transaction appears. The identifiers of newly identified affected global transactions are sent to all related sites for further assessment.

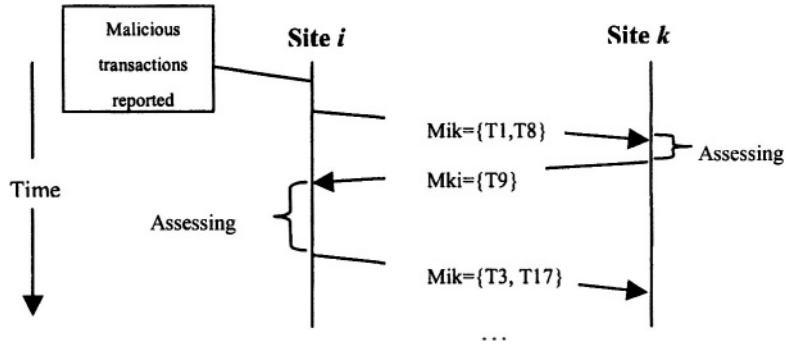

Figure 2: Time line of assessment process of peer-to-peer model

For further clarification of the situation, consider the following scenario, as depicted in Figure 2. A site *i* identifies global transactions, T_1, T_8, and T_{20} as affected. The site manager then discovers that T_1 and T_8 have sub-transactions executed at site *k* and T_{20} has its subtransactions executed at sites other than site *k*. For simplicity, we only consider communications between site *i* and site *k*. Site *i* then sends a list of affected transactions $\{T_1, T_8\}$ to site *k*. We denote this message as $M_{ik} = \{T_1, T_8\}$, where M_{ik} is the message sent from site *i* to site *k* and the message contains a list of affected global transaction identifiers, $\{T_1, T_8\}$ in this case. Site *i* does the same for other sites if necessary. After site *k* receives this message, it performs assessment and identifies several global transactions including T_9 as affected. After learning that transaction T_9 has a sub-transaction executed at site *i,* site *k* sends a message $(M_{ki} = \{T_9\})$ to site *i.* After site *i* receives this message from site *k* and performs assessment, it does not identify any new sub-transaction of any global transaction that is dependent on T_9 (or T_9's sub-transactions). Hence site *i* does not send any new message based on the previously received message to site *k.* Some time later, site *i* identifies global transactions T_3 and T_{17} are affected (based on information received from other sites) and both these transactions have sub-transactions executed at site *k.* Once again, site *i* sends a message containing T_3 and T_{17} to site *k.* The process continues until no affected transactions found at site *i* or *k.*

The advantages of this model include: (1) the process is fully distributed, so, every site executes the same algorithm and has a balanced data

processing and communication load; (2) there is no single point of failure; (3) the processing load is distributed among local sites and the damage assessment time can be minimum. One of the disadvantages of this model is the relatively large amount of network traffic. Another disadvantage is that the synchronization process for this approach can be complicated. This model is best suited for databases with small number of distributed sites. In order for this model to work effectively any two sites in the system should communicate with each other directly.

3.2 Centralized model

This model requires one coordinator for the purpose of damage assessment. The coordinator can be produced through a voting process when the distributed database system starts up. A site which is most likely to hold the coordinator (called coordinator site) should have the following specifications: (1) located at the most "convenient" place in the system in term of network distance; (2) equipped with super-processing abilities; (3) connected with all other sites by high speed network links; (4) backup made by at least one other machine in the same site in case of system failure. Any site with highest combined features of these characters will be given preference to host the coordinator.

In addition to coordinating with the coordinator, each site manager is also responsible for scanning the local log. Each site manager exclusively communicates with the coordinator. The centralized model can be further divided into three sub models: receive and forward model, local dependency graph model, and central graph repository model. These models are described next.

3.2.1 Receive and forward model

In this model, the coordinator receives and forwards messages appropriately. The assessment process is iterative and recursive. The coordinator site keeps information about each global transaction to determine where its sub-transactions are executed.

During the damage assessment and recovery procedure, each site manager sends a list of global affected transaction identifiers to the coordinator, which further locates the sites where sub-transactions of any of these affected transactions were executed. Then the coordinator sends each identified site a list, which contains the affected global transaction identifiers, which have sub-transactions executed at that site. When a site manager receives this list, it uses this information to further assess if any more transactions at its own site have been affected. It does so by scanning

the local log to find any transactions, which are dependent upon any of the affected transactions received from the coordinator. If no more global transactions have been newly detected as affected, it sends a "clear" message to the coordinator, which means the received affected transaction list has not lead to any new detection. On the other hand, if some global transactions have been detected as affected based on the received information, their identifiers are sent to the coordinator. In any case, the newly identified affected local transaction list is kept locally without sending it to the coordinator.

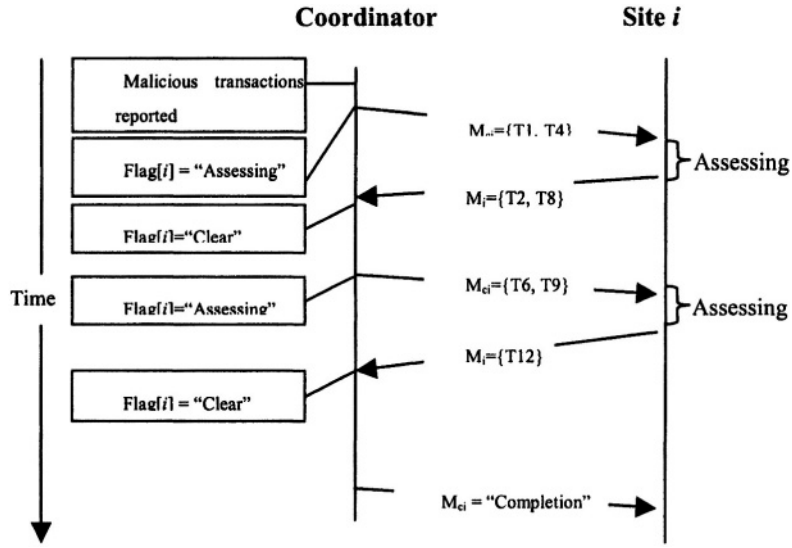

Figure 3: Time line of assessment process of receive and forward model

The coordinator keeps an array of flags for each site to keep track of each site status for the purpose of assessment. Initially, each flag is set to "clear". When the coordinator identifies some affected global transactions having sub-transactions running at a site, it sends a message to that site and resets that site's corresponding flag to "assessing". When it receives a reply from that site, the coordinator resets the flag for that site to "clear". When all the site flags are "clear", the assessment process is over.

Figure 3 gives an example of the basic assessment process. Initially, the coordinator is reported that sub-transactions of T_1 and T_4 are affected at site *i*. So it sets Flag[*i*] to "assessing" and sends this list to site *i,* which further assesses and identifies global transactions T_2 and T_8 as affected. Hence site *i* sends this information to the coordinator. When the coordinator receives the

reported $M_i = \{T_2, T_8\}$, it resets Flag[*i*] to "clear". Then the coordinator identifies which sites have executed sub-transactions of T_2 and T_8, and sends the corresponding lists to those sites. Later on, the coordinator receives information that T_6 and T_9 are affected. It then sends a list containing T_6 and T_9 to site *i* since T_6 and T_9 have sub-transactions executed at site *i*. Site *i* receives this information and assesses. This process continues until all site flags are set to "clear".

This model has the best scalability of all models presented in this paper and can be used in large-scale distributed databases. Message passing between the coordinator and each site can be intensive given a database with a large number of distributed sites but the processing burden is light at the coordinator site. The damage assessment efficiency is largely dominated by the communication effectiveness between the coordinator and each local site.

3.2.2 Local dependency graph model

When a site manager reports malicious global transactions, the coordinator sends a request message to every site, *i*, asking for the local transaction dependency graph, G_i, maintained at each site. The coordinator creates a bad transaction list to store the affected transaction identifiers, say, $\{B_1, B_2, \ldots, B_n\}$ for each site. The coordinator builds the global transaction dependency graph using the local graphs $G_1, G_2, \ldots, G_n$. Several methods have been already developed to construct such a global graph. Some of these methods, their advantages, and drawbacks can be found in [2, 5, 6].

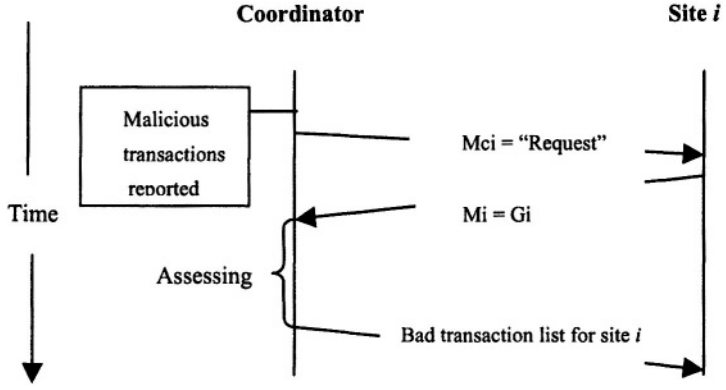

Figure 4: Time line of assessment process of local dependency graph model

Figure 4 depicts the assessment procedure of the local dependency graph model. In this model, the coordinator carries heavy loads since it performs the assessment based on the built global transaction dependency graph. The coordinator maintains a global affected transaction list. If any transaction is dependent on any of the transactions in the affected list, the former transaction is determined as affected and its global identifier is added to the affected list if it is a sub-transaction of a global transaction. After all transactions in the global dependency graph are evaluated, the assessment process becomes over. The coordinator sends a list of affected transactions, $B_1, B_2, \ldots, B_n$ (or their sub-transactions) to the corresponding sites.

It is crucial for the coordinator to be equipped with super-processing abilities, i.e., multiple-processors running in parallel. Only inter-process communications in the coordinator site are necessary if multiple processes are employed. Hence, the reduced amount of network communications (which could be among sites with far away from each other) is one of the advantages of this model. Another positive side of this model is that the assessment procedure is not recursive at each site. One of its disadvantages is the overload on the coordinator machine. This may cause processing delays in a system with a large number of sites. Another disadvantage is that a site can't start repair process until it receives the final damaged transaction list (including both global and local transactions).

This model is best suited for middle-scale distributed database systems with limited network bandwidth capability. Message passing between the coordinator and each local site is minimum and processing burden for each site is very light. The coordinator processing ability largely influences the assessment time but the initial time to build a global dependency graph can cause delays after malicious transactions are reported since the coordinator has to ask each site for transaction dependency graph to be sent to it.

3.2.3 Central graph repository model

This model has each site send its local transaction dependency graph to the coordinator periodically. The coordinator stores each graph and updates the corresponding dependency graph only when it receives any change to the graph from a site. Every site transmits to the coordinator only changes to its local dependency graph since the last transmission. After receiving an update from a site, the coordinator sends an acknowledgement back to that site to make it aware that the coordinator has received the update. Then the site can simply delete the copy of the sent message. For any message without acknowledgement, each site keeps a copy in a buffer since that could be used for further investigation (this update may not be used by the coordinator since it has already started its assessment work). Having

received the malicious transaction identifier list from a site (sites), the coordinator scans each dependency graph recursively and identifies any affected transactions based on the local dependency graphs $G_1, G_2, \ldots, G_n$. It records a list for each site, which contains the damaged sub-transactions executing on that site. Figure 5 shows the procedure for damage assessment between the coordinator and site *i*.

In this model, the coordinator keeps the local transaction dependency graphs separately instead of building the global transaction dependency graph unlike the local dependency graph model. Next, we analyze both positive and negative aspects of both of these methods.

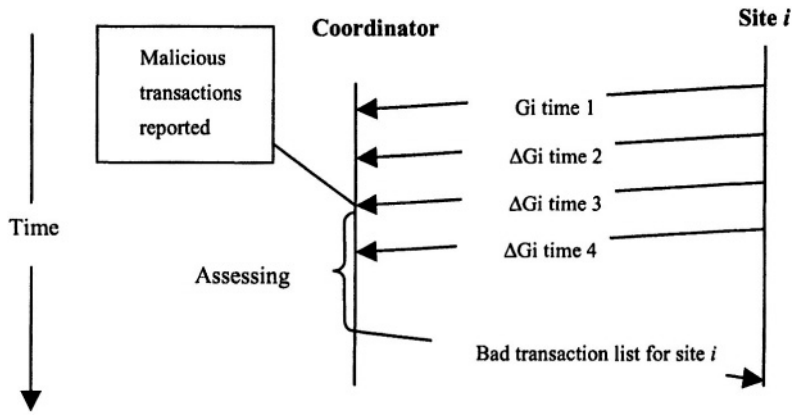

Figure 5: Basic assessment process of central graph repository model

If the coordinator wishes to keep a damaged transaction list for each site, then the local dependency graph model is more desired. If a global dependency graph was built, any newly identified damaged transactions had to be further investigated to see which sites they are associated with. Separating each local dependency graph would directly put any newly identified transaction Ids into the site list immediately after scanning the local dependency graph. Another reason to avoid building a global dependency graph is time and efforts required to build the global dependency graph itself. But building a global dependency graph has its own advantages. The coordinator does not need to jump back and forth among multiple local dependency graphs while it is performing damage assessment. It requires only one pass of the whole dependency graph when the assessment process is performed. Hence, the coordinator in the local

dependency graph model as described in the previous section uses a global dependency graph. In this model, the coordinator ignores any new updates it receives from a site during the assessment process. It just employs the most recent information before starting the assessment work. Since the coordinator does not always have the most recent information about each local transaction dependency graph, the identified affected transaction list for some sites may not be the final results. Hence, the coordinator puts a timestamp for each list in order for each site to do further investigation if necessary. The options for further work are broad. Basically, any of the models discussed in this paper could be employed to do further investigation since the last update of the local dependency graph.

This model works best for middle to large-scale distributed databases with coordinators equipped with a cluster of processors, which can achieve parallelism in processing multiple local dependency graphs. The communication between the coordinator and each local site may vary from moderate to intensive depending on frequency of updates to local dependency graphs being sent to the coordinator by each site.

4. SYCHRONIZATION ISSUES

For any model with recursion among multiple objects, a synchronization mechanism needs to be employed to deal with the order of information flow. In this section we discuss how to incorporate synchronization mechanisms to our models.

4.1 Message serial numbers in the receive and forward model

Figure 6 shows the synchronization mechanism using serial numbers to keep track of information flow. Each message is tagged with a serial number sent out from each site (including the coordinating site). If the coordinator sends out M_{ci-1} to site *i* and shortly it sends out another M_{ci-2} (after receiving a message from other site and performing the assessment), it should receive two replies from site *i*, M_{i-1} and M_{i-2}, which corresponds to M_{ci-1} and M_{ci-2} respectively (where Mc_{i-1} and Mc_{i-2} are messages 1 and 2 sent from the coordinator to site *i*. M_{i-1} M_{i-1} are messages 1 and 2 replied by site *i* to the coordinator). In the basic model shown earlier without synchronization mechanism, at time point *A* in Figure 6, after the coordinator receives M_{i-1}, based on the basic model, it would set the status flag of site as "clear". Actually, the flag should not be set to "clear" since M_{ci-2} is yet to be received by the coordinator from site *i*. The status flag can

be set to "clear" at the coordinator site only after the replay for the last M_{ci} is received. So, for the last M_{ci} received by site *i*, if there are no more affected global transactions identified, site *i* sends a "clear" message to the coordinator (M_i = "clear").

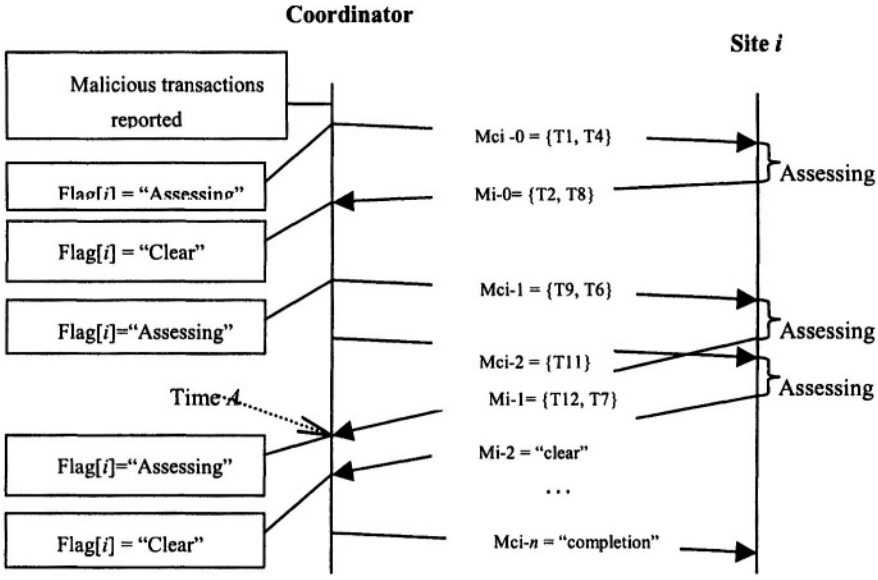

Figure 6: Synchronized assessment process for receive and forward model

If a new message arrives from a site *i* while site *k* is processing another message from site *i,* a window buffer similar to the sliding window in TCP protocol kept in site *k* can hold the second message to wait for the next iteration of the assessment process. By using this window, each site should keep track of the responses it has received and the ones that are pending. Hence, the flow-control mechanism can assure that a previous message with a smaller serial number is processed before processing a later message with a larger serial number.

4.2 Status messages among multiple sites in the peer-to-peer model

In the peer-to-peer model, synchronization helps detect termination for the assessment algorithm. Each site keeps track of the status for every other site in order to know precisely when the assessment is over. The value of the

flag reads either "assessing", meaning the site is doing assessment work, or "clear", meaning it has detected all affected transactions based on the current information. Whenever a site receives a bad transaction list, it sends an "assessing" message to all other sites. After it finishes the assessment work based on the received message, it sends a "clear" message to all other sites to let them know it is done. If a site does not receives a bad transaction list for a certain time, it sends out a "clear" message again to let its peers know that it is clear. Every site also sends an "assessing" message to all other sites telling them which of them should perform assessment if the sender identifies some global transactions as affected. When a site finds that all the flags of all sites read "clear", it knows that the assessment is over at every site.

5. CONCLUSION

In this paper, we discussed centralized and peer-to-peer models for distributed database damage assessment. These models are different from those for the centralized database systems. In the context of distributed database systems, collaborations among multiple sites (directly or indirectly) are crucial. This is determined by the nature of global transactions, which have sub-transactions executed at multiple sites. Identification of affected transactions is the burden of each site manger, which is responsible for scanning the log and checking for the transaction dependency graphs. In the peer-to-peer model, each site communicates with the coordinator extensively and the coordinator acts to receive and forward corresponding messages to the appropriate sites. The centralized model puts much of the burden of damage discovery on the coordinator site and the requirement for message transmissions between each site and the coordinator is reduced.

Acknowledgement

This work has been supported in part by US AFOSR under grant F49620-01-10346. The authors are thankful to Dr. Robert. L. Herklotz for his support, which made this work possible.

References

[1] P. Ammann, S. Jajodia, C. D. McCollum, and B. Blaustein, "Surviving Information Privacy, p. 164-174, Oakland, CA, May 1997.

[2] E. Bernstien, V. Hadzilacos, and N. Goodman, "Concurrency Control and Recovery in Database Systems". Addison-Wesley, Reading, MA, 1987.

[3] R. Graubart, L. Schlipper, and C. McCollum, "Defending Database Management Systems against Information Warfare Attacks". Technical report, The MITRE Corporation, 1996.

[4] S. Jajodia, C. D. McCollum, and P. Amman, "Trusted Recovery". Communications of the ACM, 42(7), pp. 71-75, July 1999.

[5] H. Korth, E. Levy, and A. Silberschatz, "A Formal Approach to Recovery by Compensating Transaction". In Proceedings of the 16th VLDB Conference, Brisbane, Australia, 1990.

[6] Scott D. Lathrop, Gregory J. Conti, Daniel J. Ragsdale, "Information Warfare in the Trenches". Security education and Critical Infrastructures, January 2003.

[7] Peng Liu, Xu Hao, "Efficient Damage Assessment and Repair in Resilient Distributed Database Systems". IFIP TC11/WG11.3 Fifteenth Annual Working Conference on Database and Security, July 15-18, 2001.

[8] P. Liu, P. Ammann, and S. Jajodia, "Rewriting Histories: Recovering from Malicious Transactions". Distributed and Parallel Databases, 8(1), pp. 7-40, January 2000.

[9] B. Panda and J. Giordano, "Reconstructing the Database After Electronic Attacks". Database Security XII: Status and Prospects, S. Jajodia (editor), Kluwer Academic Publishers, 1999.

[10] P. Ragothaman, and B. Panda, "Modeling and Analyzing Transaction Logging Protocols for Effective Damage Assessment", In Proceedings of the 16th Annual IFIP WG 11.3 Working Conference on Data and Application Security, King's College, University of Cambridge, UK, July 2002.

[11] R. Sobhan and B. Panda, "Reorganization of Database Log for Information Warfare Data Recovery". In Proceedings of the 15th Annual IFIP WG 11.3 Working Conference on Database and Application Security, Niagara on the Lake, Ontario, Canada, July 15-18, 2001.