A Qualified Analysis of Synchronous Non-Conflicting Retrieval Line Amassing Protocols for Fault-Tolerant Mobile Distributed Systems

Yogendra Kumar Katiyar, Research Scholar, Department of ECE, Sunrise University, Alwar, Rajasthan, INDIA.
Email: yogendra.katiyar@gmail.com
Dr. Ram Mohan Singh Bhadoria, Associate Professor, Department of ECE,Sunrise University, Alwar, Rajasthan, INDIA

ABSTRACT

Fault Tolerance Techniques enable setups to implement tasks in the presence of faults. A repossession-pinpoint (checkpoint) is a native state of an undertaking saved on steady storage. While dealing with Mobile Distributed setups, we come across some concerns like: mobility, low bandwidth of cordless passages and lack of steady storage on mobile nodules, disconnections, restricted battery power and high miscarriage rate of mobile nodules. These concerns make customary NRL-amassing (Non-conflicting Retrieval Line amassing) techniques designed for Distributed setups unsuitable for Mobile environments. To stockpile a repossession-pinpoint, a Nm_Nod (Mobile Nodule) must transfer a large amount of repossession-pinpoint data to its native Nm_Sp_St over the cordless network. Since the cordless network has low bandwidth and Nm_Nods have low computation power, all-undertaking NRL-amassing will left-over the scarce resources of the mobile setup on every repossession-pinpoint. Bottommost-undertaking coherent NRL-amassing is a preferred approach for mobile distributed setups. In this paper, we discuss various existing bottommost-undertaking NRL-amassing blueprints for mobile distributed setups.

Keywords: checkpointing; parallel & distributed computing; rollback retrieval; fault-tolerant systems

1. INTRODUCTION

Parallel computing with clusters of workstations is being used extensively as they are cost-effective and scalable, and can meet the demands of high-performance computing. Increase in the number of components in such setups increases the miscarriage probability. It is, thus, necessary to examine both hardware and software solutions to ensure fault tolerance of such parallel computers. To provide fault tolerance, it is essential to understand the nature of the faults that occur in these setups. There are mainly two kinds of faults: enduring and makeshift. Permanent faults are caused by enduring damage to one or more components and makeshift faults are caused by changes in environmental conditions. Permanent faults can be rectified by repair or replacement of components. Makeshift faults remain for a short duration of time and are difficult to detect and deal with. Hence it is necessary to provide fault tolerance particularly for makeshift failures in parallel computers. Fault-tolerant techniques enable a setup to implement tasks in the presence of faults. It is easier and more cost effective to provide software fault tolerance solutions than hardware solutions to cope with makeshift failures [1, 2].

Native repossession-pinpoint is the saved state of an undertaking at An undertaking or at a given instance. Global repossession-pinpoint is a collection of native repossession-pinpoints, one from each undertaking. A global state is said to be “unfailing” if it contains no conflicting dispatch; i.e., a dispatch whose receive event is recorded, but its send event is lost. A transit dispatch is a dispatch whose send event has been recorded by the sending undertaking but whose receive event has not been recorded by the getting undertaking [1, 7].

The problem of stockpiling a repossession-pinpoint in a dispatch passing distributed setup is quite complex because any arbitrary set of repossession-pinpoints cannot be used for retrieval [9]. This is due to the fact that the set of repossession-pinpoints used for retrieval must form a unfailing global state.

NRL-amassing is classified into following categories: Asynchronous/Un-coherent NRL-amassing

- Synchronous/Coherent NRL-amassing
- Quasi-Synchronous or Communication-induced NRL-amassing
- Missive Logging based NRL-amassing

The problem of stockpiling a repossession-pinpoint in a dispatch passing distributed setup is quite complex because any arbitrary set of repossession-pinpoints cannot be used for retrieval. This is due to the fact that the set of repossession-pinpoints used for retrieval must form a unfailing global state.

1. ASYNCHRONOUS NRL-AMASSING

Under the asynchronous approach, repossession-pinpoints at each undertaking are taken independently without any orchestration among the undertakings. Because of absence of orchestration, there is no guarantee that a set of native repossession-pinpoints taken will be a unfailing set of repossession-pinpoints. Thus, a retrieval blueprint must search for the most recent unfailing set of repossession-pinpoints before it can begin retrieval [8], [9].

1.2 SYNCHRONOUS / COHERENT NRL-AMASSING

In coherent or synchronous NRL-amassing, undertakings coordinate their native NRL-amassing actions such that the set of all recent repossession-pinpoints in the setup is guaranteed to be unfailing [add reference list……]. In case of a fault, every undertaking restarts from its most recent enduring/enduring repossession-pinpoint. Hence, this approach simplifies retrieval and it does not suffer from domino-effect. Furthermore, coherent NRL-amassing requires each undertaking to maintain only one enduring repossession-pinpoint on steady storage, reducing storage overhead and eliminating the need for garbage collection. Its main disadvantage is the large latency involved in output commit [15].

A straightforward approach to coordinate NRL-amassing is to block dispatches while the NRL-amassing undertaking is executing. A coordinator stockpiles a repossession-pinpoint and disseminates a requisition dispatch to all undertakings, asking them to stockpile a repossession-pinpoint. When an undertaking collects a dispatch, it stops its execution, flushes all the communication passages, stockpiles a conditional-enduring repossession-pinpoint, and sends an acknowledgement dispatch back to the coordinator. After the coordinator collects acknowledgement from all undertakings, it disseminates a commit dispatch that completes the two stage NRL-amassing blueprint. After getting the commit dispatch, each undertaking collects the old enduring repossession-pinpoint and makes the conditional-enduring repossession-pinpoint enduring. The undertaking is then free to resume execution and exchange dispatches with other undertakings. The coherent NRL-amassing blueprints can also be classified into following two categories: bottommost-undertaking and all undertaking blueprints. In all-undertaking coherent NRL-amassing blueprints, every undertaking is compelled to stockpile its repossession-pinpoint in an instigation [7], [8]. In bottommost-undertaking blueprints, bottommost work together ing undertakings are compelled to stockpile their repossession-pinpoints in an instigation.

The coherent NRL-amassing blueprints can be classified into two types: stalling and non-stalling. In stalling blueprints, as mentioned above, some stalling of undertakings takes place during NRL-amassing [4]. In non-stalling blueprints, no stalling of undertakings is compelled for NRL-amassing [5].

In a centralized blueprint like Chandy-lamport [7], there is one nodule which always pledges the repossession-pinpoints and coordinates the participating nodules. The disadvantage of a centralized blueprint is that all nodules must begin repossession-pinpoints whenever the

centralized nodule decides to repossession-pinpoint. Nodules can be given autonomy in initiating repossession-pinpoints by allowing any nodule in the setup to begin repossession-pinpoints.

1.3. MOBILE DISTRIBUTED COMPUTING SETUP

A mobile computing setup consists of many Nm_Nods and relatively fewer Nm_Sp_Sts. The distributed computation we consider consists of n spatially separated sequential undertakings denoted by P_0, P_1, ..., P_{n-1}, running on fail-stop Nm_Nods or on Nm_Sp_Sts. Each Nm_Nod or Nm_Sp_St has one undertaking running on it. The static network provides steadfast, sequenced delivery of dispatches between any two Nm_Sp_Sts, with arbitrary dispatch latency. The cordless network within a cell ensures FIFO delivery of dispatches between an Nm_Sp_St and a native Nm_Nod, i.e., there exists a FIFO channel from a Nm_Nod to its native Nm_Sp_St, and another FIFO channel from the Nm_Sp_St to the Nm_Nod. If a Nm_Nod did not leave the cell, then every dispatch directed to it from the native Nm_Sp_St would be received in the classification in which they are directed [2].

To send a dispatch from a Nm_Nod h_1 to another Nm_Nod h_2, h_1 first sends the dispatch to its native Nm_Sp_St over the cordless network. This Nm_Sp_St then forwards the dispatch to the native Nm_Sp_St of h_2 which forwards it to h_2 over its native cordless network. Since the location of a Nm_Nod within the network is neither fixed nor universally known to the network, because, a Nm_Nod switches cells frequently. The native Nm_Sp_St of h_1 needs to first determine the Nm_Sp_St that currently serves h_2. This is essentially the problem that has been tackled through a variety of routing blueprints at the network layer. Hence, the cost incurred for routing and delivering a dispatch to a Nm_Nod, varies with the specific routing blueprint being used. Here, we have assumed that any dispatch destined for a Nm_Nod incurs a fixed search cost [2].

1.4 NRL-AMASSING CONCERNS IN MOBILE DISTRIBUTED COMPUTING SETUPS

The existence of mobile nodules in a distributed setup introduces new concerns that need proper handling while designing an NRL-amassing blueprint for such setups. These concerns are mobility, disconnections, finite power source, vulnerable to physical damage, lack of steady storage etc. [2, 19, 20]. The location of a Nm_Nod within the network, as represented by its current native Nm_Sp_St, changes with time. NRL-amassing blueprints that send control dispatches to Nm_Nods, will need to first locate the Nm_Nod within the network, and thereby incur a search overhead [2]. Due to vulnerability of mobile computers to catastrophic failures, disk storage of a Nm_Nod is not acceptably steady for storing dispatch logs or native repossession-pinpoints. NRL-amassing blueprints must therefore, rely on an alternative steady repository for a Mob-Nod's native repossession-pinpoint [2]. Disconnections of one or more Nm_Nods should not prevent recording the global state of an application executing on Nm_Nods. It should be noted that cessation of a Nm_Nod is a voluntary undertaking, and recurrent disconnections of Nm_Nods is an expected feature of the mobile computing environments [2]. The battery at the Nm_Nod has restricted life. To save energy, the Nm_Nod can power down individual components during periods of low activity [2]. This strategy is referred to as the doze mode undertaking. A Nm_Nod in doze mode is awakened on getting a dispatch. Therefore, energy conservation and low bandwidth constraints require the NRL-amassing blueprints to abate the number of orchestration dispatches and the number of repossession-pinpoints.

2. SOME BOTTOMMOST-IMPLEMENTATION COHERENT NRL-AMASSING BLUEPRINTS FOR MOBILE DISTRIBUTED SETUPS

2.1 GUOHONG CAO AND MUKESH SINGHAL BLUEPRINT [5]Cao and Singhal achieved non-intrusiveness in the bottommost-undertaking blueprint by introducing the concept

of mutable repossession-pinpoints. In their blueprint, initiator, say P_{in}, sends the repossession-pinpoint requisition to any undertaking, say P_j, only if P_{in} collects m from P_j in the current CI. P_j stockpiles its conditional-enduring repossession-pinpoint if P_j has directed m to P_{in} in the current CI; otherwise, P_j concludes that the repossession-pinpoint requisition is a useless one. Similarly, when P_j stockpiles its conditional-enduring repossession-pinpoint, it propagates the repossession-pinpoint requisition to other undertakings. This undertaking is continued till the repossession-pinpoint requisition reaches all the undertakings on which the initiator transitively depends and a NRL-amassing tree is formed. During NRL-amassing, if P_i collects m from P_j such that P_j has taken some repossession-pinpoint in the current instigation before sending m, P_i may be forced to stockpile a repossession-pinpoint, called mutable repossession-pinpoint. If P_i is not in the bottommost set, its mutable repossession-pinpoint is useless and is discarded on commit. The huge data structure MR [4] is also attached with the repossession-pinpoint requisitions to condense the number of useless repossession-pinpoint requisitions. The rejoinder from each undertaking is directed directly to initiator.

2.2 KUMAR AND KUMAR BLUEPRINT [14]

They recommended a blueprint which is based on keeping track of direct dependencies of undertakings. Initiator Nm_Sp_St collects the direct dependency vectors of all undertakings, works out the conditional-enduring bottommost set (bottommost set or its subset), and sends the repossession-pinpoint requisition along with the conditional-enduring bottommost set to all Nm_Sp_Sts. This step is taken to condense the time to gather the coherent repossession-pinpoint. It will also condense the number of useless repossession-pinpoints and the stalling of the undertakings. Suppose, during the execution of the NRL-amassing blueprint, *Pi* stockpiles its repossession-pinpoint and sends *m* to *Pj*. *Pj* collects *m* such that it has not taken its repossession-pinpoint for the current instigation and it does not know whether it will get the repossession-pinpoint requisition. If *Pj* stockpiles its repossession-pinpoint after undertaking ing *m*, *m* will become conflicting. To evade such conflicting dispatches, they propose the following technique. If *Pj* has directed at bottommost one dispatch to An undertaking, say *Pk* and *Pk* is in the conditional-enduring bottommost set, there is a good probability that *Pj* will get the repossession-pinpoint requisition. Therefore, *Pj* stockpiles its induced repossession-pinpoint before undertaking ing *m*. An induced repossession-pinpoint is like the mutable repossession-pinpoint [18]. In this case, most probably, *Pj* will get the repossession-pinpoint requisition and its induced repossession-pinpoint will be converted into enduring one. There is a less probability that *Pj* will not get the repossession-pinpoint requisition and its induced repossession-pinpoint will be discarded. Alternatively, if there is not a good probability that *Pj* will get the repossession-pinpoint requisition, *Pj* buffers *m* till it stockpiles its repossession-pinpoint or collects the commit dispatch. They have tried to minimize the number of useless repossession-pinpoints and stalling of the undertaking by using the probabilistic approach and buffering selective dispatches at the receiver end. Exact dependencies among undertakings are maintained. It abolishes the useless repossession-pinpoint requisitions and reduces the number of duplicate repossession-pinpoint requisitions.

2.3 SILVA AND SILVA BLUEPRINT [15]

The authors recommended all undertaking coherent NRL-amassing blueprint for distributed setups. The non-intrusiveness during NRL-amassing is achieved by attaching monotonically expanding repossession-pinpoint number along with computational dispatch. When An undertaking collects a computational dispatch with the high repossession-pinpoint number, it stockpiles its repossession-pinpoint before undertaking ing the dispatch. When it gathers the

repossession-pinpoint requisition from the initiator, it ignores the same. If each undertaking of the distributed program is endorsed to begin the repossession-pinpoint undertaking, the network may be flooded with control dispatches and undertaking might left-over their time making unnecessary repossession-pinpoints. To evade this, Silva and Silva give the key to begin repossession-pinpoint blueprint to one undertaking. The repossession-pinpoint event is triggered periodically by a native timer blueprint. When this timer expires, the initiator undertaking the repossession-pinpoint state of undertaking running in the machine and dictate all the others to stockpile repossession-pinpoint by sending a disseminate dispatch. The interval between adjacent repossession-pinpoints is called repossession-pinpoint interval.

2.4 KOO-TOUEG'S BLUEPRINT [10]

They recommended a bottommost undertaking stalling NRL-amassing blueprint for distributed setups. The blueprint consists of two phases. During the first stage, the repossession-pinpoint initiator identifies all undertakings with which it has communicated since the last repossession-pinpoint and sends them a requisition. Upon getting the requisition, each undertaking in turn identifies all undertaking it has communicated with since the last repossession-pinpoint and sends them a requisition, and so on, until no more undertakings can be identified. During the second stage, all undertaking identified in the first stage stockpile a repossession-pinpoint. The result is an unfailing repossession-pinpoint that involves only the participating undertakings. In this blueprint, after an undertaking stockpile a repossession-pinpoint, it cannot send any dispatch until the second stage terminates successfully, although getting dispatches after the repossession-pinpoint is permissible.

2.5 COA AND SINGHAL STALLING BLUEPRINT [4]

They presented a bottommost undertaking NRL-amassing blueprint in which, the dependency information is recorded by a Boolean vector. This blueprint is a two-stage blueprint and hoards two kinds of repossession-pinpoint on the steady storage. In the first stage, the initiator sends a requisition to all undertakings to send their dependency vector. On getting the requisition, each undertaking sends its dependency vector. Having received all the dependency vectors, the initiator constructs an N*N dependency matrix with one row per undertaking, represented by the dependency vector of the undertaking, based on the dependency matrix, the initiator can locally calculate all the undertaking on which the initiator transitively depend. After the initiator finds all the undertaking that need to stockpile their repossession-pinpoints, it adds them to the set S_{forced} and directs them to stockpile repossession-pinpoints. Any undertaking getting a repossession-pinpoint requisition stockpiles the repossession-pinpoint and sends a reply. The undertaking must be blocked after getting the dependency vectors requisition and resumes its computation after getting a repossession-pinpoint requisition.

2.6 PRAKASH-SINGHAL BLUEPRINT [16]

It was the first blueprint to combine these two approaches i.e., bottommost undertaking and non-impeding. More specifically, it dictates only a bottommost number of undertakings to stockpile repossession-pinpoints and does not obstruct the underlying mensuration during NRL-amassing. Prakash Singhal blueprint [16] dictates only part of undertakings to stockpile repossession-pinpoints, the snpsht_class_no of some undertakings may be out-of-date, and may not be able to evade discrepancies. It attempts to solve this delinquent by having each undertaking maintains an array to save the snpsht_class_no, where chkpt_seq_noi[i] has been the expected snpsht_class_no of Pi. Note that Pi's chkpt_seq_noj[i] may be divergent from Pj 's chkpt_seq_noj [i] if there is no communication between Pi and Pj for several repossession-pinpoint intervals. By using

snpsht_class_no and the founder identification number, they claim that their non-impeding blueprint can evade discrepancies and curtail the number of repossession-pinpoints during NRL-amassing.

2.7. P. KUMAR'S HYBRID BLUEPRINT [13]

In bottommost-undertaking coherent NRL-amassing, some undertakings may not repossession-pinpoint for several repossession-pinpoint initiations. In the case of a retrieval after a fault, such undertakings may rollback to far earlier repossession-pinpoint ed state and thus may cause greater defeat of computation. In all-undertaking coherent NRL-amassing, the retrieval line is advanced for all undertakings but the NRL-amassing overhead may be exceedingly high. To optimize both matrices, the NRL-amassing overhead and the defeat of computation on retrieval, P.Kumar recommended a hybrid NRL-amassing blueprint, wherein an all-undertaking coherent repossession-pinpoint is taken after the execution of bottommost-undertaking coherent NRL-amassing blueprint for a fixed number of times. Thus, the Mobile nodules with low activity or in doze mode undertaking may not be disturbed in the case of bottommost-undertaking NRL-amassing and the retrieval line is advanced for each undertaking after an all-undertaking repossession-pinpoint. Additionally, he tried to abate the information attached onto each computation dispatch. For bottommost-undertaking NRL-amassing, he designed a stalling blueprint, where no useless repossession-pinpoints are taken and a determination has been made to optimize the stalling of undertakings. He recommended to delay selective dispatches at the receiver end. By doing so, undertakings are endorsed to implement their normal computation, send dispatches, and partially receive them during their stalling period. The recommended bottommost-undertaking stalling blueprint dictates zero useless repossession-pinpoints at the cost of very inconsequential stalling.

2.8 Kumar & Khunteta [17] Blueprint

They have recommended a bottommost undertaking coherent NRL-amassing blueprint for mobile distributed setup, where no useless repossession-pinpoints are taken and a determination is made to abate the stalling of undertakings. The number of undertakings that stockpile repossession-pinpoints is abated to evade awakening of Nm_Nods in doze mode of undertaking and thrashing of Nm_Nods with NRL-amassing activity. Further, it hoards restricted battery life of Nm_Nods and low bandwidth of cordless passages. We have used the concept of delaying selective dispatches at the receiver end only during the NRL-amassing period. By using this technique, only selective undertakings are blocked for a short duration and undertakings are endorsed to do their normal reckonings and send dispatches in the stalling period. They captured the transitive dependencies during the normal execution. The Z-dependencies are well taken care of in this blueprint. They also evaded collecting dependency vectors of all undertakings to compute the bottommost set. Thus, the recommended blueprint is simultaneously able to condense the useless repossession-pinpoints to zero and tries to optimize the stalling of undertakings at very less cost of maintaining exact dependencies among undertakings and attaching repossession-pinpoint classification numbers and dependency vectors onto normal computation dispatches.

2.9 Garg and Kumar Blueprint [18]

They have recommended a nonstalling coherent NRL-amassing blueprint for mobile distributed setups, where only bottommost number of undertakings stockpiles enduring repossession-pinpoints. They have reduced the dispatch complexity in comparison to Cao-Singhal blueprint [5], while keeping the number of useless repossession-pinpoints unchanged. The recommended blueprint is designed to impose low memory and computation overheads on Nm_Nods and low

communication overheads on cordless passages. A Nm_Nod can remain disconnected for an arbitrary period without affecting NRL-amassing activity. They address the concerns like: failures during NRL-amassing, disconnections, maintaining exact dependencies among undertakings, and contemporaneous initiations. They also attempt to abate the defeat of NRL-amassing determination if some undertaking backfires to stockpile its repossession-pinpoint in the first stage but it will increase the orchestration overhead.

CONCLUSION

The existence of mobile nodules in a distributed setup introduces new concerns that need proper handling while designing a NRL-amassing blueprint for such setups. These concerns are mobility, disconnections, finite power source, vulnerable to physical damage, lack of steady storage, etc. The new concerns make customary NRL-amassing techniques unsuitable to repossession-pinpoint mobile distributed setups. A good NRL-amassing blueprint for mobile distributed setups should have low memory overheads on Nm_Nods, low overheads on cordless passages and should evade awakening of a Nm_Nod in doze mode undertaking. The cessation of a Nm_Nod should not lead to infinite wait state. The blueprint should be non-intrusive and should dictate bottommost number of undertakings to stockpile their native repossession-pinpoints .

Bottommost-undertaking coherent NRL-amassing is a suitable approach to introduce fault tolerance in mobile distributed setups transparently. This approach is domino-free, requires at most two repossession-pinpoints of an undertaking on steady storage, and dictates only a bottommost number of undertakings to repossession-pinpoint. It may require stalling of undertakings, extra orchestration dispatches, attaching of some information along with computation dispatches, or stockpiling some useless repossession-pinpoints.

In this paper we have given introductory concepts related to NRL-amassing in mobile distributed setups. We also gave an analysis of various bottommost-undertaking NRL-amassing blueprints especially designed for mobile distributed setups.

REFERENCES

[1] Singhal, M, N G Shivratri, “Advanced concepts in Operating Systems, Tata Mc Graw Hill, 1994.

[2] Acharya A., “Structuring Distributed Algorithms and Services for networks with Mobile Hosts”, Ph.D. Thesis, Rutgers University, 1995.

[3] Cao G. and Singhal M., “On coordinated checkpointing in Distributed Systems”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no.12, pp. 1213-1225, Dec 1998.

[4] Cao G. and Singhal M., “On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile Computing Systems,” *Proceedings of International Conference on Parallel Processing*, pp. 37-44, August 1998.

[5] Cao G. and Singhal M., “Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing systems,” *IEEE Transaction On Parallel and Distributed Systems*, vol. 12, no. 2, pp. 157-172, February 2001.

[6] Cao G. and Singhal M., “Checkpointing with Mutable Checkpoints”, *Theoretical Computer Science*, 290(2003), pp. 1127-1148.

[7] Chandy K. M. and Lamport L., “Distributed Snapshots: Determining Global State of Distributed Systems,” *ACM Transaction on Computing Systems*, vol. 3, No. 1, pp. 63-75, February 1985.

[8] Elnozahy E.N., Alvisi L., Wang Y.M. and Johnson D.B., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 375-408, 2002.

[9] Kalaiselvi, S., Rajaraman, V., "A Survey of Checkpointing Algorithms for Parallel and Distributed Systems", *Sadhna,* Vol. 25, Part 5, October 2000, pp. 489-510.

[10] Koo R. and Toueg S., "Checkpointing and Roll-Back Recovery for Distributed Systems," *IEEE Trans. on Software Engineering*, vol. 13, no. 1, pp. 23-31, January 1987.

[11] Parveen Kumar, Lalit Kumar, R K Chauhan, V K Gupta "A Non-Intrusive Minimum Process Synchronous Checkpointing Protocol for Mobile Distributed Systems" *Proceedings of IEEE ICPWC-2005*, January 2005.

[12] Pushpendra Singh, Gilbert Cabillic, "A Checkpointing Algorithm for Mobile Computing Environment", *LNCS, No. 2775*, pp 65-74, 2003.

[13] Parveen Kumar, "A Low-Cost Hybrid Coordinated Checkpointing Protocol for Mobile Distributed Systems", *Mobile Information Systems* [An International Journal from IOS Press, Netherlands] pp 13-32, Vol. 4, No. 1, 2007.

[14]. Lalit Kumar, Parveen Kumar "A Synchronous Checkpointing Protocol for Mobile Distributed Systems: A Probabilistic Approach", *International Journal of Information and Computer Security* [An International Journal from Inderscience Publishers, USA], pp 298-314, Vol. 3 No. 1, 2007.

[15]. Silva L, Silva J 1992 Global checkpointing for distributed programs. *Proc. IEEE 11th Symp. On Reliable Distributed Syst*. pp 155-162.

[16] R. Prakash and M. Singhal. "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems". *IEEE Trans. on Parallel and Distributed System***,** pages 1035-1048, Oct. 1996.

[17] Praveen Kumar, Ajay Khunteta, "A Minimum-Process Coordinated Checkpointing Protocol for Mobile Distributed System" International Journal of Computer Science issues, Vol. 7, Issue 3, 2010

[18] Rachit Garg, Praveen Kumar, "A Nonblocking Coordinated Checkpointing Algorithm for Mobile Computing Systems", International Journal of Computer Science issues, Vol. 7, Issue 3, 2010

[19] Deverpalli Raghu, Parveen Kumar," A Crossbreed Orchestrated Temporary Snapshot based Amalgamated coordinated Consistent Recovery Line Accumulation Protocol for Mobile Distributed Systems", International Journal of Electrical Engineering and Technology" Vol. 11, Issue 9, Nov 2020, pp.225-238.

[20] Praveen Choudhary, Parveen Kumar," Effectual Minimum-Process Consistent Recovery Line Etiquette for Mobile Ad hoc Networks", International Journal of Electrical Engineering and Technology" Vol. 11, Issue 7, Nov 2020, pp.31-37.