International Journal of Engineering, Science and Mathematics
Vol. 6 Issue 8, December 2017 (Special Issue)
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Novel Notion of Hybrid Checkpointing Strategy in Mobile Distributed Computing System

Anup Patnaik*
M. Vamsi Krishna**

Abstract

Keywords:

Fault tolerance;
Coordinated checkpoint;
Message logging;
Mobile distributed system;
Clustercommunication model.

This paper proposes an original and efficient hybrid snapshot (Checkpoint) protocol on cluster basedmobile distributed system. Preceding, mobile distributed system has acquired massive attention in recent years due to its design and features able to monitor and control applications running on the wireless network efficiently through the fault tolerance methods that offer additionally consistency and reliability to the flow. Further, previous research papers on this distributed paradigm have not considered all the potential overheads incurred, therefore our designed algorithm has made an effort on this which is the fine blend of coordinated and message logging protocols provides failure free operation and simple recovery process on failure. Based on this hybrid mechanism we propose Novel Notion of Hybrid Checkpointing Strategy (NNHCS) that not only improves the performance of the system with increasing availability of mobile hosts (MHs), less interruption to MHs due to single phase coordination but also decreases overheads of coordination message and piggybacking information. As per our knowledge, this is the first algorithm will address issues of cluster communication model based computing system with well integration of mobile agents. The simulation results obtained from MATLAB tool suggests that performance of the proposed prototype is independent of communication and computation overhead with increase of MHs.

Author correspondence:

Research Scholar,Department of Computer Science and Engineering, Centurion University of Technology and Management, Odisha, India

1. Introduction

In classical prospective, checkpointing and recovery scheme implementation is limited to static host connected in high speed wired network i.e. static distributed systems but with the growing demand on wireless network due to its several benefits have brought the new era of mobile distributed environment with the advent of wireless technology. Besides, wireless network connection in the mobile environment as compared to the wired distributed environment [1- 4] is more fragile and prone to failure, hence the challenges are quite high toretain the consistency and reliability of the system for its long execution time. Basically, the prototypes being used in trivial distributed system can't be applied directly to mobile distributed system due to presence ofdifferent constraints on this system such as Mobility of MHs, limited battery power on MH, limited wireless bandwidth, noisy wireless environment, frequent handoff, and limited stable storage in turn make checkpointing algorithm less effective. Ample of researches [5-7] have been done from different corners on checkpointing strategy to tolerate faults and continue execution without much acute losses, eventually reached a conclusion that failure free execution and failure recovery operation using checkpoint strategy definitely put up better performance considering incurred overheads and latencies than other varied approaches. When we

* Research Scholar, Department of Computer Science and Engineering, Centurion University of Technology and Management, Odisha, India

**HOD, Department of Computer Science and Engineering, Centurion University of Technology and Management, Odisha, India are considering checkpointing approaches for our proposed prototype then the first and foremost key facts, those come into mind initially are as how frequently to allow checkpoints to seize processor state, storage location of different kinds of checkpoint, contents of checkpointing, coordination/application messages and some incurred overheads. Moreover, two main checkpointing approaches distinctly prevail in this system, namely Disk based and Diskless checkpointing approaches. Many researches on these approaches stated that each has its own virtues and drawbacks than the other one, therefore selection of right approach depends on architecture, requirement, and criterion of the system.

In the Kohafi et al. [8], it was proposed that Diskless checkpoint achieves better performance but less reliable with additional memory and processors overhead whereas, Disk based checkpoint attains better reliability for highly critical application with the compromise on the performance side. This summary from the previous empirical analysis between the former and later approaches paves us the way to choose the disk based system here, since we are considering critical mobile distributed system in which the need of reliability in this current juncture is must to have (Figure 1).

Before delve into more on checkpointing strategies [3, 9-11] some of the common definitions used in fault tolerant though checkpoint province explicated below, we also used those definition in this paper.

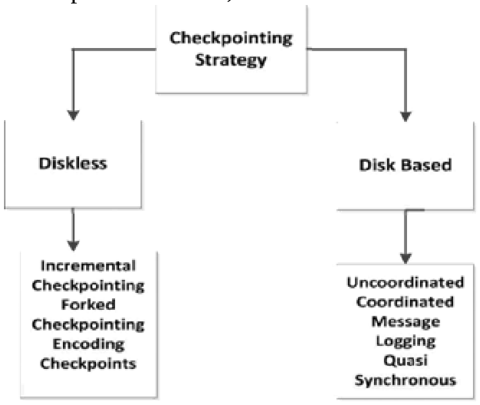
Checkpointing Strategy
Diskless
Disk Based
Incremental Checkpointing Forked Checkpointing Encoding Checkpoints
Uncoordinated Coordinated Message Logging Quasi Synchronous

Figure 1.*Classification of Checkpointing Approaches*

Definitions

Lamport's (happens before' relation): If X and Y are two events occurring in the same process and if X occurs before Y, then X →Y and If A is the event of sending a message and B is the event of receiving the same message in another process then, A → B.

Watchdog Interval: It’s assumed to be fixed interval for the process to checkpoint its state both for intra and inter cluster communication. During this specific period, all the processes inside and outside cluster makes their checkpoints consistent w.r.to each other.

Lost Message: Send event of message recorded by the sender process but not recorded by the receiver process on that watchdog interval (Figure 2).

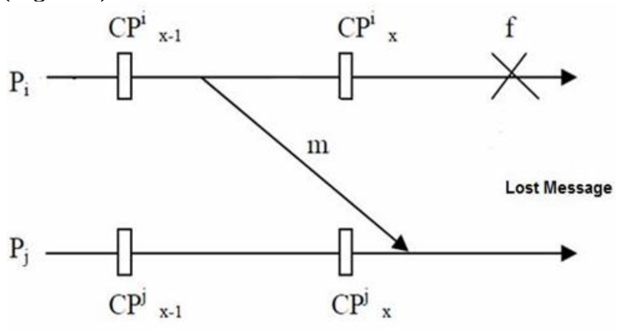

Figure 2.*Example of Lost Message*

Orphan Message: Send event of message not recorded by the sender process but recorded by the receiverprocess on a watchdog interval (Figure 3).

Delayed Message: Send event of message recorded by the sender process but still in communication channel,not yet received by the receiver process (Figure 4).

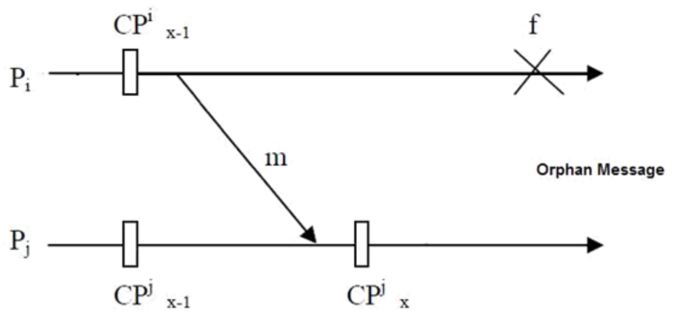

Figure 3. *Example of Orphan Message*

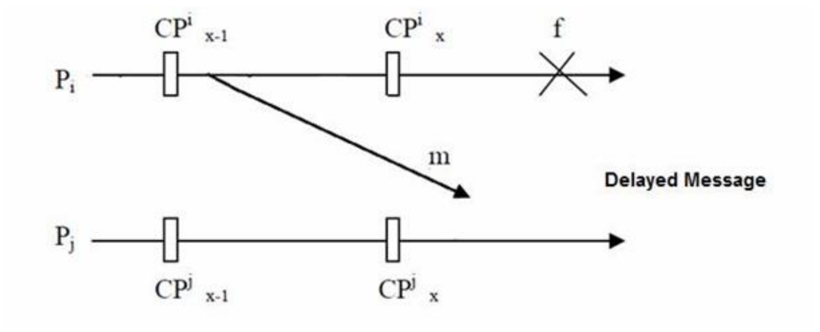

Figure 4.*Example of Delayed Message*

Internal Events: Events happened inside the same process.
External Events: Events happened w.r.to other processes from a process
Application/Computation Messages: Generated by processes to communicate among each other
Coordination/Control Messages: Process initiator communicates with other process to establish thecoordination
Tentative/Induced Checkpoint: Checkpoint taken when coordination messages by initiator reaches to theprocesses (Figure 5).
Forced Checkpoint: Checkpoint taken in response to the computation messages to avoid orphan messages(Figure 5).
Hand off: Handing of disconnected MHs from one location to other location for connection.
Domino Effect: Cascaded rollback of processes to initial state
Avalanche Effect: If any checkpointing schemes continue without termination among the processes based onthe dependency correlation in any checkpointing interval then it raises the phenomenon of avalanche effect

As our proposed prototype based on usage of stable storage, hence we will focus on disk based distributed checkpointing here and also skim through some existing similar checkpointing schemes.

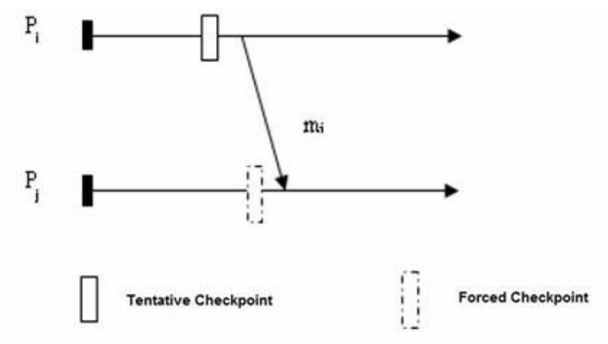

Figure 5.*Example of Tentative and Forced Checkpoint*

Uncoordinated Checkpointing

This scheme does not coordinate the checkpoints between processes rather use specialized algorithms to determine the set of consistent checkpoints on recovery and keeps most or all of the generated checkpoints on stable storage since it is not known until restart which set of those checkpoints are required for failure recovery operation. We may suffer from having useless checkpoints in stable storage and also from domino effect due to inconsistency, all the processes may get started from the very initial stage. But they do not suffer the synchronization overhead during failure-free operation as with coordinated protocols.

Biswas and Neogy [12] designed checkpointing protocol for combination movement pattern using Handoff checkpointing and Periodic checkpointing concepts. Mobility, movement patterns and Handoffs of MHs are key strength behind this checkpointing and recovery protocol. Based on movement pattern, Handoff based checkpointing for intercell, Periodic based checkpoint for intracell and again Handoff Based checkpointing for combination pattern but it was observed that handoff threshold reached longer time than intercell. This prototype only addressed one type of failure scenario i.e. disconnection of MHs with planned and unplanned means, concept of Migration checkpoint proposed for planned disconnection which ensures not incurring any delay in checkpointing. The performance results show that checkpoint initiator only sends to dependent MHs and MHs save migration checkpoint before planned disconnection only

Coordinated Checkpointing

In this scheme, all the processes are coordinated to form the global consistent state from the local checkpoints, hence in case of recovery all processes can start from the recent checkpoints stored on global state. By resorting to this scheme, fault tolerance is free from the domino effect, reducing storage overhead as only one permanent checkpoint flushed to stable storage in any checkpoint interval and avoid the need of external garbage collection algorithm to free up the storage space. Apart from these advantages, several pitfalls are also arose such as it blocks communications while checkpointing process executes, decides whether all processes or minimum processes participate in synchronization leads to many incurred overheads, and high latency to output storage. Below highlighted some of the algorithms falling under this category,

Cao and Singal [13] proved that no coordinated checkpoint is non –blocking and forces minimum number of processes to take checkpoints. But later different research papers stated some minimum process efficient checkpoints algorithms. Chandy/Lamport [14] algorithm allows a process to checkpoint once it has received a special marker token from every process in the system. This method requires FIFO communication channels between all processes to form a consistent state. It is assumed that once the marker has been received all other messages on that channel can be delayed by the system until the checkpoint is established.

Koo- Toueg’s [15] proposed a minimum process two phase blocking check pointing algorithm for distributed systems. During the first phase, the checkpoint initiator identifies all process with which it has communicated since the last checkpoint and subsequently sends them a request. Upon receiving the request, each process in turn identifies all processes it has communicated since their last checkpoint and sends them a request, and so on, until no more processes can be identified. During the second phase, all processes identified in the first phase take a checkpoint. The result is a consistent checkpoint that involves only the participating processes.Inthis protocol, after a process takes a checkpoint, it cannot send any message until the second phase terminates successfully.

Awasthi and Kumar [16] proposed a minimum process coordinated checkpointing protocol for mobile distributed systems, where the number of useless checkpoints and the blocking of processes are reduced using the probabilistic approach and by computing the tentative minimum set in the beginning. This algorithm is the first one to combine blocking and non-blocking scheme in one algorithm. Basu et al. [9] addresses the problem of overhead involved in taking checkpoints and time to recover from a failure in an attempt to make a tradeoff between efficiency and reliability on the existing algorithms. This algorithm certainly reduces the recovery cost after failure of a mobile host with the suitable consideration of Handoff threshold. The checkpointing decision depends not only on the mobility factor but also on the location distance between the mobile support stations. Moreover the stationary checkpointing scheme makes the algorithm robust against unnecessary checkpoints while mobile host remains stationary for a considerable amount of time.

Kumar et al. [17] address the mobile computing system issues those holding back the effectiveness of traditional checkpointing algorithms and presented non-intrusive minimum process synchronous checkpointing protocol brings the optimization on minimum number of tentative checkpoints, number of useless forced checkpoints and message overheads for recovery and failure free execution. In order to create the minimum process checkpointing scheme, the initiator process collects the direct dependency vectors of all the processes, computes the minimum set and sends the checkpoint request along with the minimum set to the relevant processes. This algorithm adopted by crashing the height of the checkpointing tree and by reducing the uncertainty period of processes to minimize the useless induced checkpoints. No blocking of the processes and no need to send huge data structure for the checkpoint request as it ensures the initiator process keeps exact CSN of all the processes i.e. most recent permanent checkpoint.

Cao and Singhal [18] introduced the mutable checkpoint to design efficient checkpointing algorithm which forces the minimum number of processes to take their checkpoints. However, advantage of taking mutable checkpoint is to get rid of transferring large amount of data to the stable storage at MSSs over the wireless network. The performance of this algorithm is measured in terms of number of tentative checkpoint, output commit delay and system message overhead, especially it seems this algorithm needs more system messages than [19] but crucial point is overhead of system messages much smaller than the overhead of checkpoints on the stable storage. Simulation result shows that this algorithm significantly reduces the message overhead as compared to [20] and blocking time is 0 whereas [20] has needless blocking time which downgrades system performance. Algorithm features suit for heterogeneous environment with high reliability requirement, failure prone MHs, frequent checkpointing situations.

Quasi-Synchronous Checkpointing

Unlikely coordinated checkpointing, this checkpoint doesn't exchange any special kind of explicit coordination message rather it sends all the relevant control information piggybacking with application message that is called internal synchronization process. Its strategy provides the scheme free from domino effect using notion of basic and forced checkpoints. Tongchit and Manivannan [7] proposed communication induced checkpointing protocol where the recovery happens asynchronously through the selective message logging strategy to handle the messages lost in rollback. Main strength of having communication induced checkpoint algorithm allows the processes to take checkpoint asynchronously and reduces the number of useless checkpoints being taken additional checkpoints at appropriate time. Processes can take checkpoints at any time.i.e. message received by a process can start to take forced checkpoint before processing and protocol only piggybacks checkpoint sequence number with each control messages. Message replying strategy adopted in this scheme takes care of delayed, lost and duplicate messages. No orphan messages whatsoever as processes roll back to consistent global state. Selective message logging scheme comes into play for the messages lost during recovery and disconnection, reduces message logging overhead as MSS selectively log messages that need to be replayed after the rollback. Gass and Gupta [21] disclosed one efficient communication induced checkpointing algorithm along with different forced checkpoint concept than the usual forced checkpoint. Process fires two events for receiving of application message and taking checkpoint simultaneously. This algorithm relies concept of finding out the set of globally consistent checkpoint periodically shows many advantages such as avoid synchronization delay, recovery process is simple as process aware of GCC to rollback after recovery and GCC having both local and forced checkpoints, the amount of rollback reduces.

Luo and Manivannan [22] presented both Basic and Advanced FINE checkpointing algorithm shown better performance than other [23]. This Basic FINE (Fully Informed aNd Efficient checkpointing algorithm) using couple of data structures for ZCF property and TDE-Timestamp ensures that it takes better checkpoint inducing decision and same time decrease the overhead of piggybacked information. Subsequently Advanced FINE checkpointing having better/stronger checkpoint inducing condition further reduces overhead of piggybacked information.

Message Logging checkpointing

It is used to combine checkpointing with logging of non-deterministic events so that logging and replaying the determinants of message could be used during recovery, further a process can attain the pre failure state in spite of unavailable of latest checkpointed state. Also, this log based roll back recovery enables the processes to recover beyond the most recent consistent checkpoint set which is very much useful when we consider our prototype communication between the inter-clusters. This logging scheme is classified into pessimistic, Optimistic and causal logging. Chowdhury and Neogy [24] presented a rollback recovery algorithm based on independent checkpointing and message logging. In this algorithm mobile agents are used to manage the message logs and checkpoints. When mobile node goes far away from its latest checkpoint then the agents manage to move the checkpoint and message logs stored in distant Mobile Service Stations to other stations. Thus recovery time of a mobile node will never exceed a certain threshold. Logging of messages ensures that only one checkpoint is needed to be stored in persistent storage. As independent checkpointing is used, no synchronization messages are needed to be exchanged hence saving network bandwidth. Since the applications for the wireless network typically exchanges lesser number of smaller messages as compared to its wired counterpart, the message log is not too large for the MSS buffers to store. This algorithm shows better result as compared other algorithms which use mobile agents in synchronous checkpointing for either Hamiltonian topology or any topology. Finally, we would like to provide the summarized tabular form of checkpointing algorithms measured through different parameters as following (Table 1).

Table 1.*Summarization of checkpointing algorithms measured through different parameters*

Schemes Parameters	Uncoordinated Checkpoint	Coordinate Checkpoint	Quasi Synchronous Checkpoint	Pessimistic Logging Checkpoint	Other Loggings Checkpoint

Domino Effect	Yes	No	No	No	Yes
Avalanche Effect	No	Yes	Yes	No	No
Process Autonomy	High	Low	Average	High	High
Orphan Process	Yes	No	Yes	No	Yes
Concurrent Checkpointing & Multiple Failures Handling	No	Yes	Yes	No	No
Recovery	Independently faulty process restored	Rollback to last set of checkpoints	Needed large number of forced checkpoint	Last set of Checkpoints	Multiple checkpoints considered to protect orphan condition
Extent of Potential Rollback	Unbounded	Global Checkpoint State	Several Checkpoints	Last Checkpoint	Last Checkpoint or Previous checkpoints
Failure Recovery Computation	Complex	Simple	Complex	Simple	Complex
Overheads	Checkpointing overheads, Large storage	Message overheads for coordination	Useless checkpoints, Piggybacking of huge information and storage overheads	Message logs overheads, Performance overheads	Message logs overheads, Complex recovery overhead

1.1.Problem Formulation

Our objective is to address transient failures of distributed mobile system with cluster communication model which stays for short duration time during operation. Being fault in the system, fault tolerance techniques allow the system to rerun from specific state that's why it's highly recommended and worthwhile tohave one effective and efficient technique. The new proposed prototype has the following characteristics such as Enforces Minimum Processes, Single Phase Synchronization, Non intrusiveness, and Non-Blocking checkpointing Protocol. Unlike other blocking and, high checkpointing/message overhead algorithms, our
checkpointing algorithm is free from all such pitfalls as our research efforts are directed towards to overcome the overhead involved on common practices followed on checkpoint approach. Design of our NNHCA algorithm is motivated by using coordinated and pessimistic message logging schemes can handle both the intra-cluster and inter-cluster failure free and failure recovery operations. The crux behind our prototype design is to provide the communication and computation overhead independent from with increase of MHs, minimum coordination overhead and also no useless checkpoints at whatsoever.

This manuscript has organized into six different sections based on the assumptions, design & implementation of algorithm, empirical analysis and future directions. In Section 2, System Model and Background, in Section 3 Detail elaboration of Novel Notion of Hybrid Checkpointing Protocol (NNHCS) protocol, Sections 4 more precisely defines Proof of Correctness with Lemma and Theorems, Section 5 Empirical Analysis on Performance with other parallel algorithms, and Section 6 provides conclusion remarks.

2. System Model and Background

We assume there are set of $\mathbf{p_1, p_2\ldots,p_n}$ processes concurrently running on different cluster groups as $\mathbf{C_1,C_2\ldots C_k}$of mobile distributed system and each cluster has n number of processes. Inside cluster, one nodepresent as cluster prime primarily facilitates different events happenings inside, other than cluster primes each node represents a MH(mobile host) running with one process. Execution of the process is modeled with three kinds of events i.e. Send Event, Receive Event, Internal Event. Lamport's happened before relationship can be established between send and receive event for the same message. Every message reaches to the receiver through the cluster prime.

Mobility of MHs, limited battery power on MH, limited wireless bandwidth, noisy wireless environment, frequent handoff, and limited stable storage on MH induce challenging problems for consistency and reliability to all types of mobile computing systems through fault-tolerance. Further, to establish the coordination among the processes running on multiprocessor systems either it will go through the message passing or share memory of system, but in our prototype the only way for processes to communicate with each other is by passing messages through a reliable, asynchronous channel with unpredictable but having a finite transmission delays, hence no messages will be lost and order of the message is also preserved in the channel. The message delivery of the wired links and wireless links follow strict FIFO communication, no other algorithm shows the urge of message order in case of the wired communication. Since our prototype focuses on the both the intra-cluster and inter-cluster communication, we are assuming the FIFO communication channel for inside cluster and outside cluster.

We also assumed that computation involved in the system, adopted to piece-wise deterministic model in which a process always produces the same sequence of states in its execution for the same sequence of message-receiving events. Each process runs inside the processor, no bound to their speed and also in case of failure, it is assumed that processes fail according to the fail-stop model where a process is permanently stopped, happened due to a crash without any additional erroneous outcomes. Also system failures are transient faults and independent, it means that process at same point of time after the recovery won't fail again. In our proposed prototype, mobile distributed application follows cluster based communication model where the system typically grouped into different cluster groups, every cluster group has cluster prime which is of rich of resources and handles all communication inside and outside its boundary. No separate communication transport protocol used in cluster prime to deliver intra cluster message and inter cluster messages, rather follows the order it receives messages. We assumed here that cluster prime nominates the process to be the initiator for the checkpointing request event under its supervision. Our NNHCA algorithm has to handle both the intra-cluster and inter-cluster failure free & recovery operations. For all the inter cluster operation we preferred to use logging scheme than checkpointing scheme, since later scheme involves large amount re-computation and also coordination message overhead to get started from consistent state during recovery.

Several researches earlier have shown that pessimistic logging has simple log based recovery scheme than the optimistic logging due to the later logging scheme has substantial chances falling under domino effect, hence process has to keep multiple checkpoints for failure recovery operation which downsizes system performance. Especially, no orphan message generated at any point of time, no additional efforts needed for the ordering of messages delivered in pre failure state possible with pessimistic logging scheme which stays one of core viewpoint behind this prototype design. Two core techniques are wrapped in designing our proposed prototype Novel Notion of Hybrid Checkpointing Strategy (NNHCA) explained below.

2.1. Coordinated checkpoint protocol for intra cluster

It is of Minimum Processes, Single Phase Synchronization, Non Intrusiveness, and Non-Blocking checkpointing Protocol.

Observation 1: P_i, $1 \leq i \leq n$, is checkpoint dependent on $P_j$if one of the following conditions holds true:
(1) Some process P_i, $i \leq i \leq n$, takes a checkpoint before it sends out m_i;
(2) Some process P_i, $i \leq i \leq n-1$, takes a checkpoint before it receives m_i.

A global checkpoint state is constructed from the set of local checkpoints, one from each process which is said to be consistent if it does not contain orphan messages i.e. no message is recorded as received in one process and not yet recorded as sent in another process. Usually message complexity to facilitate coordination is much higher side most of the coordinated approaches, but in this algorithm we have assured to have O(Nmin) to establish coordination. Mobile agent namely(Dependency Accumulator) residing in cluster prime acts on behalf of the process initiator whose sole responsibility is to find the dependency vectors containing those processes would get checkpointing request message in later stage, so process initiator free from these computation therefore, it can preserve its resources and computations time. Also we brought new concept in this algorithm, transformation of checkpoint state one form to another form in order to prevent false checkpointing, inconsistencies, lost or delayed messages. During the transformation, earlier checkpointing state loses its relevant data structure and regains new sort of information. This prototype is single phase and minimum processes participate in the checkpointing event as Dependency Accumulator mobile agent gathers dependency information on behalf of the process initiator for direct and transitive dependents through recursive approach from the last permanent checkpoint and makes sure no avalanche effect arises, finally hands over data to process initiator to send checkpoint request to the processes listed out in Dependency Vector(simple data structure queue with direct and transitive dependent processes).

Cluster prime has the authority of selecting the process initiator inside the cluster which further gets dependency vector and sends checkpointing request message to processes. In the ideal scenario if any process gets the same checkpoint request, it will take primarily the Handshake checkpoint but there may arise some situation where checkpoint request not received yet but meantime process got application message from other processes. In such case instead of waiting for request message, it can take essential checkpoint (kind of forced checkpoint) .Later when process gets the request message from initiator, following actions will be taken place

- No further checkpoint will be taken
- Rejects the Request message from Initiator
- Transmission of essential checkpoint to handshake checkpoint as request message has reached eventually.

We assumed here that the Handshake checkpoints, not the essential checkpoints get converted to permanent checkpoint at the end of watchdog interval and later flushed to stable storage.

2.2. Pessimistic Message logging and Synchronous checkpointing recovery protocol for inter cluster

Cluster prime holds the responsibility of delivering messages to MHs i.e. every message routed through this. We have named our logging scheme as Cluster Prime based Message Logging Scheme (CPMLS) stores the determinants of message i.e. content of message and order sequence of messages before delivering to the receiver and before starting off the state intervals of non-deterministic events, therefore it is kind of pessimistic message logging algorithm. Besides, it has MessageLog_Store matrix data structure maintains the record of number of messages sent and received by the process to handle lost and delayed messages. During the inter cluster communication, determinants of non-deterministic events with relevant message information logged into cluster prime volatile memory and periodically flushed to stable storage so that on recovery, the determinants of non-deterministic events would be reused and replayed to obtain the failure free state for the failed process.

3.Novel Notion of Hybrid Checkpointing Protocol (NNHCS)

Our NNHCA proposed checkpointing protocol have the following characteristics.

- Cluster communication Based
- Distributed Checkpoint Initiator
- Selective Checkpointing and rollback
- Periodic Checkpointing
- Dynamic Checkpointing
- Cluster coordinated checkpointing

3.1. How Algorithm Works

Assumption

- All the clusters operate at same speed.
- WatchDog interval to take checkpoint is same in entire cluster groups, otherwise impossible to handle the synchronization among the clusters when dependency exists through the message passing

Steps to Algorithm Development

1. Cluster Prime nominates the Process Initiator and after nomination, it assigns the dependency list calculated by Dependency Accumulator mobile agent to process initiator. Dependency Accumulator mobile agent is residing inside Cluster Prime.
2. Based on the dependency list, process initiator first takes tentative checkpoint then changes its watchdog Interval, piggybacks this interval along with the request to processes (Dependency List contains minimum number of processes).
3. When checkpoint request reaches to the processes, what action a process has to take explained below and which is common for other processes as well
    - Process will take handshake checkpoint if it has not taken checkpoint yet in the current watchdog interval and also watchdog interval of initiator should be higher or same as that of process.
    - Process after taking handshake checkpoint will continue on its external events (send and receive of messages)
4. When application message reaches to a process, then the following actions to be taken by the corresponding process and these actions are common across for other processes as well.
    - Process has already taken handshake checkpoint in this interval and both process and piggyback interval information is same then it treats message as normal message.
    - Process hasn’t taken handshake checkpoint still, it means it has not received checkpoint request message as of now, hence it takes essential checkpoint to prevent inconsistencies.

- After the essential checkpoint taken, checkpoint request message has reached late to process then process discards the checkpoint request that reached to it lately and transforms essential checkpoint to handshake checkpoint.

5. Final Commit - Before watchdog interval ends, all the processes have their handshake checkpoint and no processes now exist with essential checkpoint, then process initiator's tentative and other processes handshake checkpoint converted to permanent checkpoint at the end of the watchdog interval and flushed to stable storage.
6. Process initiator interacts with processes only once while sending checkpoint request message. In this algorithm process has high autonomy to take its own checkpoint depending on varied conditions and later flush contents to stable storage. The benefit of this notion is it prevents multiple interaction, also multiple interruption to/from process initiator, gains better system performance.

Now we are explaining below how this algorithm handles messages incurred inconsistencies (Figure 6). Instead of maintaining three vectors send, receive and in- transit as in Lalit et al. [25], we have chosen one N*N matrix and each cell inside contains (mn) where m is number of send messages and n is number of receive messages. As cluster prime in this design routes messages to host, therefore it calculates this matrix every watchdog interval.

MessageLog_Store: A matrix (N*N) where Message_Store[i][i] =00(It is considered as internal event where no send and no receive message occurred. Our prototype treats m, m1, m2, m3 in different ways using this matrix.

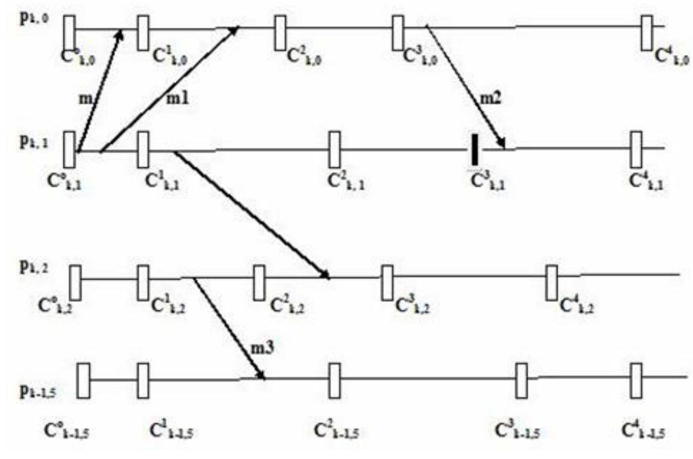

Figure 6.*Our protocol functioning*

Symbol Notations:

$p_{k,\,m}$: Process m of kthcluster
$C^{w}_{k,\,m}$:Checkpoint taken by process m of kthcluster in w watchdog interval

Secenario-1: Normal message (m)

p_1 of cluster k sends message m to same cluster process p_0. At this watchdog interval all the values in matrix zero(no receive and send message before) and this messages reaches on the same interval, hence p_0 executes the message like a normal message

Secenario-2: Delayed message (m1)

After the checkpoints ($\mathbf{C^1_{k,,0}C^1_{k,1}}$) of p_0 and p_1, the cluster prime calculates Message_Store matrix (Table 2). Clearly indicates that p_1 at this point value 20, it states 2 messages sent and no messages received and same interpretation for p_0 as well.
And the difference is Number message sent by $p_1$to p_0 – Number of message received by p_0 from p_1 i.e. 2-1 =1
Table 2.Message Log _Store matrix at watchdog interval 1(filled values only on the required cells to explainthe scenarios)

	p_0	p_1	p_2
p_0	00	01	xy
p_1	20	00	xy
p_2	xy	xy	xy

From the above achieved figures, it indicates that m1 is delayed message. Delayed messages have no right to enforce checkpoint, instead logged into stable storage along with checkpoint which may require for replaying during recovery.

Secenario-3: Orphan message (m2)

Process p_0 sends message m2 top p_1.Before any action from p_1, the matrix (Table 3) shows that there no pending messages yet to receive either by p_0 or p_1 from each other (no delayed messages).Message m2 comes from odd interval to even interval and no prior deployed message then to keep consistency (both receiving and sending should fall in same watchdog interval), it forces p_1 to take essential checkpoint, hence no orphan message

Table 3. Message_Store matrix at watchdog interval 2

	p_0	p_1	p_2
p_0	00	02	xy
p_1	20	00	xy
p_2	xy	xy	00

Secenario-4: Inter cluster message (m2)

Process p_2 of cluster k sends message m2 to process p_5 of another cluster k-1.Cluster prime maintains similar matrix for inter cluster messages communication as that of intra cluster and follows same approaches as mentioned above scenarios for handling inter cluster messages also, additionally CPMLS logs determinants of message m2 to stable storage but initially these logs are kept in volatile logs, later periodically flushed to stable storage at the end of watchdog interval. Thus pessimistic message logging neither creates any orphan message nor involves any complex process on recovery from failure.

Secenario-5: Single Phase, Non-Blocking and Non-Intrusiveness

Single Phase: Cluster prime nominated process initiator sends checkpoint request message to other processesas calculated by Dependency Accumulator (List contains minimum number of processes which received and sent messages during this watchdog interval).Interaction by initiator with processes is only once while sending checkpoint request and in our prototype processes have high autonomy to take checkpoint and later flush contents to stable storage

Non-Blocking: Our prototype doesn't suspend any process during its underlying computation. Most of theearlier algorithms either use marker message or CSN (checkpoint sequence number) to showcase the algorithm as coordinated non-blocking. As we are partially motivated by Lalit et al [40] algorithm, thus our prototype will follow same approach piggybacking even or odd interval (watchdog interval) along with each requests.

Non-Intrusiveness: Due to non-blocking nature of prototype, there is always possible chance that process mayget messages from other process running with higher checkpoint interval which leads to inconsistencies. Our prototype's design and implementation is capable of handling all kind inconsistencies as explained above on different scenarios.

3.2 Data Structures

Flag_Initiator [P] = by default false, while assigning it becomes true.
Flag_Tentative = Flag that indicates that the process has taken a tentative/induced checkpoint. Flag_Handhshake = False, by default during initialization
Flag_Essential == False, by default during initialization
m = Application message to by processed by a process
DependencyVector_Dictionary [key, values] = vector of size n
MessageLog_Store matrix = Matrix of size n*n and each cell represents number of sent and received messages of one process with respect to other process.

ClusterPrime_Disconnected = Size of n, contains the processes which are falling under disconnection

3.3 Algorithm for Inside Cluster

We use the same approach used for coordination as in [25]. Without loss of generality we can assume that all the processes start on the even WI (Watchdog interval).Piggybacking of application message is definitely light weighted, since in [25] allows single control bit along with application message indicating either odd or even watchdog interval to carry. Main difference between our proposed prototypes than algorithm [25] is in the proposed algorithm, at any point time it needs only Nmin coordination message to complete one watchdog interval and every watchdog interval is of single iteration i.e. height of checkpoint tree is always one. If we consider best case scenario of single iteration in the approach [25] then 3× N coordination message required to complete one watchdog interval which is more than this proposed prototype (Nmin). Further, Best and worst case scenarios produces same result in our model but not in [25], as it moves to complexity of n^2 level to handle coordination.

3.3.1. Checkpointing Request Procedure by Process Initiator (PI)

```
For each Cluster:

a. Cluster Prime selects the Process Initiator, assigns the Flag_Initiator[PI]= true to process initiator
b. Process initiator(PI) takes Tentative checkpoint and requests the Cluster Prime to assign the
   Dependency Vector List for direct and transitive dependencies
c. Now PI Iterates through Dependency Vector List(DependencyVector_Dictionary) after the
   assignment , Sends the checkpoint request to the processes
d. Process $P_i$ receives the checkpointing request message from
   PI First process calculates the TempStatus for the
   checkpoints
   TempStatus ← Process $P_i$ checks no checkpoint taken so far in WatchDog Interval
   (Flag_Handhshake = False and Flag_Essential= False) then return True
       If (TempStatus = True)
           Process takes handshake checkpoint and continues normal
           execution Flag_Handhshake = True
       Else if (TempStatus = False and Flag_Essential=
           True) Process rejects the checkpoint request
           Transform the Essential checkpoint to Handshake Checkpoint
e. Dependency Accumulator takes the decision if PI has to send abort message or not.
   If No message sent by PI during watch Interval after taking their Handshake
   checkpoint Then commits the handshake checkpoint to permanent checkpoint when
   timer expires Else
PI will send Abort message to all the processes in DependencyVector_Dictionary to cancel the operation
```

3.3.2. Process Sends message to other process

a. If P_i sends message to P_j
b. Updates log data structure in Cluster prime
c. Delivers to recipients

3.3.3. Process Receives message from sender process intra

```
a. $P_i$ receives coordination message
b. If not taken any checkpoint in current watchdog interval
   Takes Handshake checkpoint and continue normal operation
   Else
   Already have Essential checkpoint then Reject the coordination request
   Translate the Essential checkpoint to Handshake checkpoint
c. $P_i$ receives Application Message
d. If not taken any Handshake checkpoint in current watchdog interval
   Takes Essential checkpoint and continue normal operation
   If timer of watchdog interval expires and no abort message received from PI
   Convert Handshake checkpoint to Permanent checkpoint
e. Else if timer of watchdog interval expires and abort message received from PI
   Cancel the operation and Find the latest checkpoint from global consistent state
```

f. Else
   Execute normal execution as when message arrives

3.3.4. Recovery (Handoff Checkpointing Procedure for Disconnected MH)

a. Before disconnection of MH, it hand overs all the relevant data to Cluster Prime
b. During it disconnection, any coordination request /application message came, and then Cluster primes Clones new process with the data of disconnected MG.
c. In Reconnection, Disconnected MH sends signal to cluster prime to identify the process
d. If Disconnected MH Identified for connection then assigns the current state cloned process to it and aborts the functioning of cloned process as it is no longer requited now to function

3.4. Algorithm for Outside to Cluster

3.4.1. Process Sends message to other process (Only Application message can be sent, no coordination message)

a. If Pi sends message to Pj
   Update log data structure in current Cluster prime and Delivers to recipients to different Cluster group
b. If (application message belongs to current interval)
   Receipt cluster group delivers it to process
   Otherwise rejects the message

3.4.2.Process Receives message from sender process

a. No chance to get coordination message one cluster to another cluster, hence always application messages from other clusters to current cluster
b. IF(Process P_i receives inter cluster message)
   Process logs the Message and non-determinant events in the current process memory
c. If timer of watchdog interval expires and no abort message received from PI of Sender
   Convert Handshake checkpoint to Permanent checkpoint
d. Else if timer of watchdog interval expires and abort message received from PI of Sender Cancel the operation and Find the latest checkpoint from pessimistic message logging information to replay and rerun the determinant events with logged messages
e. Else
   Execute normal execution as when message arrives

3.4.3. Recovery (Handoff Checkpointing Procedure for Disconnected MH)

a. Before disconnection of MH, it hand overs all the relevant data to current Cluster Prime
b. During it disconnection, any coordination request /application message came, and then current Cluster Prime Clones new process assigned with data of disconnected MH.
c. In Reconnection, Disconnected MH sends signal to cluster prime of new cluster group to identify the process
d. Cluster prime of new cluster group verifies in its own disconnected list of processes and if not then broadcast the message to other cluster groups
e. If Disconnected MH identified for connection by any cluster group then assigns the current state cloned process to requested group and abort the functioning of cloned process as it is no longer requited now to function inside this group now.

4.Proof of Correctness

Theorem 1: Failure Free operation at any point of time on whole system has Global Consistent Checkpoint State.

Proof: The watchdog interval intra and inter cluster will remain the same.Without loss of generality, let'sassume that orphan messages exist in the watchdog interval which is applicable to inside and outside to a cluster group. A Process initiator of cluster group sends the checkpoint request to all the processes which it depends directly and indirectly, so processes which received the checkpoint request message have taken Handshakecheckpoint. In our prototype, the height of checkpoint tree is always one level. It not only makes sure processes of direct dependent PI part of minimum set but also covers the dependent processes of PI's direct dependentprocesses and same recursive procedure continues till the minimum set is finalized. If any process received message but not received request message, the same process can take essential checkpoint and later coverts to handshake checkpoint Hence we come to conclusion that there is no point of time where orphan message are present, so the assumption on beginning is wrong. This prototype will always provide the consistent checkpoint set.

Lemma 1: A process Pi cannot be a member of minimum set, if it has not sent or received a message in its current watchdog interval.

Proof: Without loss of generality, let's assume that$P_i$has sent some message to $P_j$and both not in part ofminimum set. As per the Lamport's definition, happens before relation exists between processes only when they send or receive application messages. This relationship on series of processes helps to find out dependency vector for process initiator. Dependency accumulator keeps those processes which followed Lamport's definition in the current watchdog interval. Surely P_i and P_j will have their place inside dependency vector list. Hence assumption is incorrect and minimum set contains {Pi, P_j }.

Lemma 2: Minimum Number of process taking checkpoints during coordination

Proof: As the theorem1 proves that participating processes which selected based received messages sincefrom last global checkpoint and mobile agent Dependency accumulator calculates the dependency vector on behalf of the process initiator. No other processes out of this dependency vector list ever receive any kind of checkpoint request message. This ensures that minimum processes always in attention during coordination.

Lemma 3: Algorithm is non-blocking and single phase coordination strategy

Proof: All the processes in the system without any wait, first it will take essential checkpoint if receivedapplication messages if no taken any checkpoint so far, then continues the normal operation. At any point of time during watchdog interval it receives the coordination message then it rejects the message as it has already holding checkpoint without any loss of any consistencies and transform the essential to handshake checkpoint. During watchdog interval only once process initiator sends checkpoint request. If the time expires processes automatically saves the checkpoints to stable storage.

Theorem 2: If a MH on one cluster hands off to another cluster during planned disconnection and then in reconnect then Inter and intra cluster groups have consistent state constructed on recovery

Proof: As theorm1 proves of global consistent checkpoint cut at any point of time over the system, itmeans to both intra and inter cluster. Without loss of generality, we will consider two clusters C_1 and C_2 and watchdog interval w1.Suppose P_1 in C_1 got disconnected, before going to disconnection mode it sends all the local data of MH to the cluster prime of C1. Further, on behalf of P1, C_1 clones new process for the disconnected MH if it receives any response from other MHs. If any message or request comes for the disconnected processes the cloned process will act right away. Now during reconnection it signals to C_2, first and foremost the C_2 cluster checks its own disconnected list if not then it broadcast messages to other cluster groups. While receiving request by C_1 from C_2 , C_1 sends recent state and all the relevant data to C_2.There is no point of time on whole system that this hand off process may go faulty which means cloned process is smart enough to maintain checkpoint state as like other processes in that cluster.

Theorem 3: The Watchdog interval is defined with finite time, later finishes at certain point where checkpoint algorithm ends successfully.

Proof: As per the Lemma 2, minimum process set is generated before sending checkpoint request. Nowthe initiator sends the request to the processes to take checkpoint. In ideal scenarios with no faulty process noticed, every checkpoint request received processes takes handshake checkpoint and carry on with their normal operation during the interval. When watchdog interval ends then transform handshake to permanent checkpoint, hence the algorithm completes successfully with no faults and terminates in the finite time period.

5.Performance Analysis

5.1. Comparison with Existing Protocol

In this section, we review previously proposed algorithms related to our checkpointing algorithm (Table 4).
Where,

N_{min}: Minimum number of processes required to take checkpoint. N_{broad}: Message broadcasted to these numbers of processes
N: total number of processes involved
N_{dep}: Number of process on which a process depends
Cost (Broadcast): Cost of broadcasting a message to all (N) processes in the system. Cost (Wireless): Cost of sending a message from one process to another process. Wckpt: The checkpointing time. This time includes the time to save the checkpoint on stable storage.

Table 4.Comparison of Different Algorithms based on Certain Crucial Criterion

Algorithms	**Blocking Time**	**Number of Checkpoints**	**Number of Coordination Phases**	**Coordinate Message Transmission Cost**	**Control Message size**
Koo-Toueg[15]	$N_{min} \times$ Wckpt	N_{min}	Three Phases	$3\times N_{min} \times Ndep\times$ (Cost (Wireless))	$3\times N_{min} \times N_{dep}$
Cao-Singhal[11]	0	N_{min}	Three Phases	$\approx 2\times N_{min} \times$ (Cost (Wireless)) +min(Nmin× (Co st (Wireless)), Cost (Broadcast))	$2\times N_{min} + N_{dep}$
Elnozahy et al. [26]	0	N	Two Phases	2× Cost (Broadcast)+ N× Cost(Wireless)	2×Nbroad +N
Kumar et al. [27]	0	N_{min}	Three Phases	$3\times N_{min} \times$ Cost (Wireless)	$3* N_{min}$
Lalit et al. [25]	0	N	Three Phases	2× Cost (Broadcast)+ N× Cost (Wireless)	$2\times N_{broad} + N$
Praveen et al. [10]	0	$N_{min} + N_{indu}$	Five Phases	N× Cost (Wireless)+2× Nmin × C(Wireless)+2× C(Broadcast)	N+2×Nmin+2 $\times N_{broad}$
Gupta et al. [6]	0	N_{min}	Single Phase	$N_{min} \times$Cost (Wireless)	N_{min}
Our Algorithm (NNHCA)	0	N_{min}	Single Phase	$N_{min} \times$ Cost (Wireless)	N_{min}

5.2.Simulation Results

We have used Matlab simulator for evaluating performance of our algorithm and the setup structure considers 10 MHs present on each clusters and total of 5 clusters, having both wireless and wired interface. Through the wired interface clusters are connected with each other with 20Mbps communication link whereas clusters and MH connected using 4Mbps.

MHs may get disconnected during the execution then its reconnection, message delivery rerouted solely controlled by cluster prime, always having 50% probability to for disconnection and 50% probability for mobility to other cluster. Primary parameters that we decided for the basis of comparison as follows

Number of useless checkpoints

The number of useless checkpoints in our case is zero at any time during the system. Transformation from essential checkpoints to induced checkpoint not only prevents the inconsistencies but also number of useless checkpoints. In case of higher message sending rate, the height of checkpoint tree remains the same but in other algorithms it increases exponentially which affects the system performance (Figure 7).

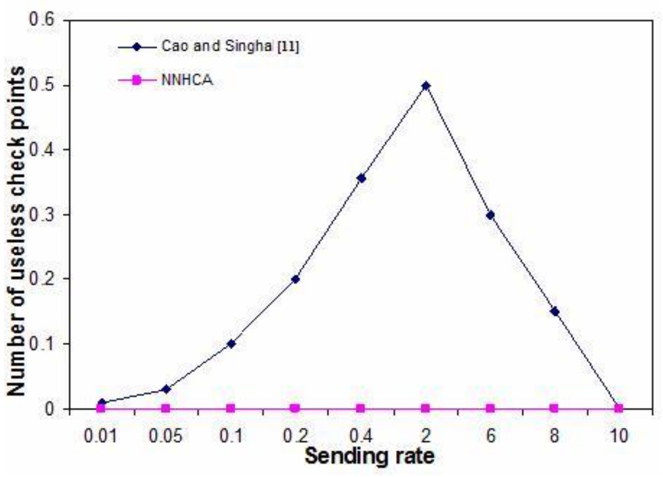

Figure 7.*Number of Useless Checkpoints vs. Sending Rate (Message Sending Rate)*

Control Messages and Coordination Transmission Cost

As compared to other algorithm, our prototype number of messages required for coordination increases gradually with increased number of processes as per results provided in the Table 2, hence outperforms other algorithm.

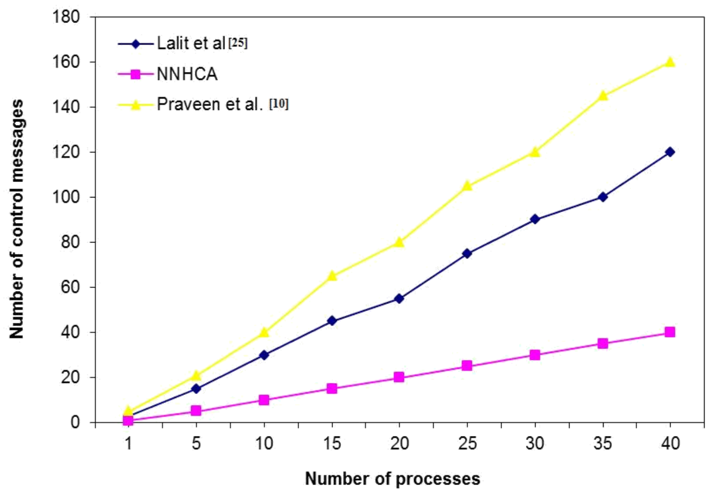

Figure 8.*Number of Control Messages vs. Number of Processes*

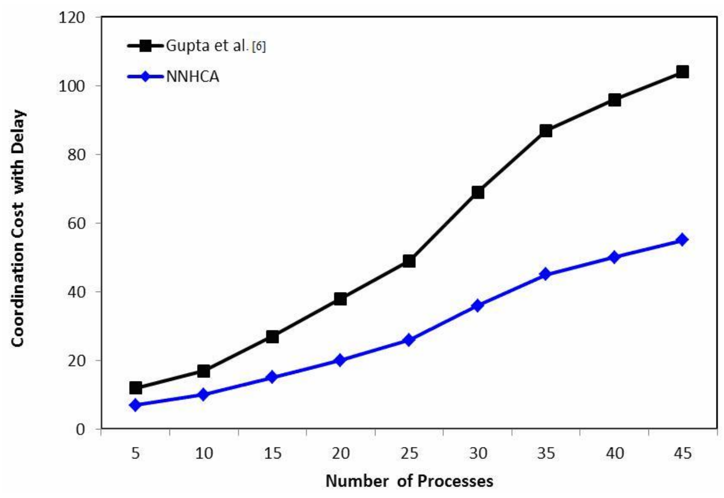

Figure 9. *Coordination Cost with Delay vs. Number of Processes*

In our algorithm, the height of checkpoint tree at any watchdog interval is one which is minimum, less

chance to have any kind of computation and control message transmission delay (Figure 8). But in case Gupta et al. [6], checkpoint tree height increases with number of processes increases leads to checkpoint request overhead and
computation of dependency overhead happens every stage hence the approach will go through significant delay than our approach even if the order of control message size for both are O(N_{min}) (Figure 9).

From the above analysis we have drawn some multi fold advantages of our proposed algorithm as compared with others as follows:

- The algorithm requires only one checkpoint per process at any time to be present in the system. The checkpoint is stored locally in the MH's local storage initially, only single phase coordination of processes to the initiator required to commit the output to the stable storage. This reduces checkpointing overhead on MH.
- The algorithm forces only a few processes to take checkpoints and to roll back on an error Recovery.
- Fine blend of coordinated checkpointing and pessimistic message logging handles failure free and failure recovery operations both in intra and inter clusters.
- High availability of disconnected MHs through new concept of cloned processes.
- Checkpointing coordination message overhead minimized in case of failure free operation

Unlike other checkpointing algorithm, this prototype doesn't enforce all the processes to take checkpoint.

6. Conclusions

Design of this protocol Novel Notion of Hybrid Checkpointing Strategy (NNHCS) Protocol is motivated by hybrid checkpoint(coordinated and message logging checkpoints) in cluster based application that require consistent global checkpoint with less coordination cost and application message overheads. This proposed algorithm is doing exceedingly well to increase the availability of MHs at any point of time, since cloned process on behalf of disconnected processes running inside the current cluster. System performance is enhanced through the high availability of MHs, lower interruption of MH due to single phase and reducing useless checkpoints. Apart from the exceeding well performance with compare to the overheads, still we are skeptical that it may bring into contention of stable storage i.e. staggered checkpoint. Further, we are carrying now on the profound investigation to shun the contention issues.

References

[1]. Awasthi, L.K., Mishra, M. and Joshi, R.C, "A Weighted Checkpointing Protocol for Mobile Distributed System," *International Journal of ad hoc and ubiquitous Computing*, vol.5(3), pp.137-149, 2010.

[2]. Kumar, L., Mishra, M. and Joshi, R.C, "Low Overhead Optimal Checkpointing for Mobile Distributed System," *Proceedings 19th International Conference on Data Engineering*, 2003, pp. 686-6883.

[3]. Kumar, P., Gupta, P. and Solanki, A. K, "Dealing with Frequent Aborts in Minimum-process CoordinatedCheckpointing Algorithm for Mobile Distributed Systems," *International Journal of Computer Applications,* vol.3(10), pp.7-12, 2010.

[4]. Kumar, P. and Khunteta, A, "Anti-message Logging based Coordinated Checkpointing Protocol forDeterministic Mobile Computing Systems," *International Journal of Computer Applications*, vol.3(1), pp.22-27, 2010.

[5]. Cao, J., Chen, Y., Zhang, K. and He, Y, "Checkpointing in Hybrid Distributed Systems. *Proc. of the 7thinternational Symposium on Parallel Architectures," Algorithms and Networks (ISPAN'04),* Hong Kong,China, 2004, pp.136-141.

[6]. Gupta, B., Rahimi, S. and Liu, Z, "A new high Performance Checkpointing Approach for Mobile ComputingSystems," *International Journal of Computer Science and Network Security,* vol.6(5B), pp.95–104, 2006.

[7]. Tantikul, T. and Manivannan, D, "A Communication-Induced Checkpointing and Asynchronous RecoveryProtocol for Mobile Computing Systems," *Proceedings of the Sixth International Conference on Parallel andDistributed Computing Applications and Technologies (PDCAT'05)*, 2005, pp. 70-74.

[8]. Kofahi, N.A., Bokhitan, S.AI. and AI-Nazer, A, "On Disk based and Diskless Checkpointing for parallel andDistributed Systems : An Empirical Analysis," *Information Technology Journal,* vol.4(4), pp.367-376, 2005.

[9]. Basu, S., Palchaudhuri, S., Podder, S. and Chakrabarty, M, "A Checkpointing and Recovery Algorithm Basedon Location Distance, Handoff and Stationary Checkpoints for Mobile Computing Systems," *InternationalConference on Advances in Recent Technologies in Communication and*

Computing, 2009, pp.56-62.

[10]. Kumar, P., Kumar, L., Chauhan, R.K. and Gupta, V. K, "A Non-Intrusive Minimum Process SynchronousCheckpointing Protocol for Mobile Distributed Systems," *Proceedings of IEEE ICPWC-2005, January 2005.*

[11]. Cao, G. and Singhal, M, "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing," IEEETrans. on Parallel and Distributed Systems, vol.12(2), pp.157-172, 2001.

[12]. Biswas, S. and Neogy, S. "A Mobility Based Checkpointing Protocol for Mobile Computimg System,"*International Journal of Computer Science and Information Technology (IJCSIT), vol.*2(1), pp.135-151, 2010.

[13]. Cao, G. and Singhal, M, "On the Impossibility of Min-Process Non-Blocking Checkpointing and An EfficientCheckpointing Algorithm for Mobile Computing Systems," *International Conference on Parallel ProcessingIEEE*, 1998, pp.37–44.

[14]. Chandy, K. M. and Lamport, L, "Distributed Snapshots, Determining Global State of Systems,"*ACMTransaction on Computing Systems*, vol.3(1), pp.63-75, 1985.

[15]. Koo, R. and Toueg, S, "Checkpointing and Rollback-Recovery for Distributed Systems," IEEE Trans. onSoftware Engineering, 1987, pp. 23-31.

[16]. Awasthi, L.K. and Kumar, P, "A synchronous checkpointing protocol for mobile distributed systems:probabilistic approach," *International Journal of Information and Computer Security*, vol.1(3), pp.298– 314, 2007.

[17]. Ssu, K.F., Yao, B., Fuchs, W. K. and Neves, N. F. "Adaptive Checkpointing with Storage Management forMobile Environments," *IEEE Transactions on reliability*, vol.48(4), pp.315-324, 1999.

[18]. Cao, G. and Singhal, M. "A New Checkpointing Approach for Mobile Computing Systems,"*IEEE Transactionson Parallel and Distributed Systems*, vol.12(2), pp.157-172,2001.

[19]. Gupta, B., Rahimi, S. and Ahmad, R. "A New Roll-Forward Checkpointing / Recovery Mechanism for ClusterFederation," *International Journal of Computer Science and Network Security*, vol.6(11), pp. 292-297, 2006.

[20]. Kim, H., Yeom, H.Y., Park, T. and Park, H. "The Cost of Checkpointing, Logging and Recovery for the MobileAgent Systems," In proceeding of: Dependable Computing, DOI:10.1109/PRDC.2002.

[21]. Gass, R.C. and Gupta, B. "An Efficient Checkpointing Scheme for Mobile Computing Systems,"*EuropeanSimulation Symposium,* 2001, pp.1-6.

[22]. Luo, Y. and Manivannan, D. "FINE: A Fully Informed and Efficient Communication-Induced CheckpointingProtocol," *IEEE Third International Conference on Systems (icons 2008)*, 2008, pp.16 -22.

[23]. Helary, J.M., Mostefaoui, A., Netzer, R. and Raynal, M. "Communication-based prevention of uselesscheckpoints in distributed computations," *Distributed Computing*, vol.13(1), pp. 29–43, 2000.

[24]. Chowdhury, C. and Neogy, S. "Checkpointing Using Mobile Agents for Mobile Computing System," *International Journal of Recent Trends in Engineering*, vol.1(2), pp.26-29,2009.

[25]. Awasthi, L.K., Mishra, M. and Joshi, R.C. "An efficient coordinated checkpointing Appraoches for DistributedComputing Systems with Reliable Channels," *International Journal of Computers and Applications*, DOI: 10.2316/Journal.202.2012.1.202-2118.

[26]. Elnozahy, E.N., Johnson, D.B. and Zwaenepoel, W. "The performance of consistent check pointing," Proc. 11thSymp. On Reliable Distributed Systems, 1992, pp. 86–95.

[27]. Kumar, S., Chauhan, R.K. and Kumar, P. "A Low Overhead minimum process Global Snapshot CollectionAlgorithm for Mobile Distributed Systems," *The International Journal of Multimedia and its Application*, vol.2(2), pp.12-30, 2010.