MODIFIED ALGORITHM FOR MOBILE HOST IN MOBILE DISTRIBUTED SYSTEM

***C.L. Mittal, **Prateek Mishra**

**Research Scholar, JJT University, Rajasthan*
***Research Scholar JJT University, Rajasthan*

ABSTRACT

Messages are generated by mobile hosts in a fixed interval of time. Number of messages generated can be increased byreducing the message inter arrival time. Message is generated and according to message type mobile host sends signal support station for further processing. The proposed the modification is to verify certain conditions after receiving the message.

As the check pointing decision is selective and hence, the total number of checkpoints is reduced. Again the algorithm has no additional control message overhead. So total time to process the message is less as compared to existing protocol where the checkpoint overhead is high.

Keywords: *Algorithm, Messages, System, Check pointing, Protocol, Modification*

EXISTING ALGORITHMS

In the existing Check pointing protocol, each host takes checkpoints independently [1]. Independently taken checkpoints are called basic checkpoints and those triggered by some message reception are called forced checkpoints. Each basic checkpoint request is generated after a fixed time interval. The checkpoint interval can vary but no condition is checked for time interval. Whenever faults occur, hosts rollback to some consistent state. This technique is although simple but may suffer from domino effect hence recovery time may be larger.

Algorithm executed at a Mobile Host (Mj):

```
1    Begin

2    Initialise network variables like self and neighbours id.

3    Create a message queue.

4    Generate the message.

5    Check (message type)

6    If (message_type = = Normal)
     {        Examine whether it is send or not.
```

```
        If (send)
                Send the message to particular host.
        else
                Receive the message. }

7       If (message type= =Checkpoint Request)
        {       execute procedure checkpoint }

8       If (messge_type= = Failure signal)
        {       send failure signal to MSS.
                Receive the message from MSS to suspend operation. }

9       If (message_type= = Recovery signal)
        {        Send signal to MSS that I`m going to recover.
        Wait,
                Collect all information for restarting by MSS.
        Restart from the last consistent point. }

10      Goto step 4

11      End
```

Algorithm executed at a Mobile Host for checkpoint:

```
1   Begin procedure checkpoint
2   Send checkpoint request to MS.
3   {Record its local state, messages of mobile host.
    Transfers local state, messages to MSS.}
4     Receive acknowledgement from MSS.
5     End procedure checkpoint.
```

The mobile support station is the backbone of mobile computing system. Each mobile host can talkto other hosts only through MSS. Mobile support station takes the checkpoint of the particular mobilehost when checkpoint request is generated. MSS sends signals like suspend operation wake up and also gives the consistent point from where mobile and fixed station can start their execution after failure and recovery.

Algorithm executed at a mobile Support Station:

```
1    Begin
2    For all mobile hosts and fixed station in the system
3    Receive the messages
4    Check (message _type)
5    If (message type = =Normal)
         {    Forward the message to destination host.  }
6      If (message _ type = = Checkpoint Request )
         {        Take checkpoint for source mobile host.
         Update the message and state log for mobile host
         Send acknowledgement to source mobile host.   }
7    If (message _type = = Failure Signal)
         {    Send signal to suspend working to source host.
         And all the hosts which are communicating with faulty host        }
8    If (message type= =Recovery Signal)
         { Scan the message and state log.Make a consistent global state.
         Send rollback signal to host and,
         dependent hosts to their last consistent checkpoint state.        }
9  Goto step 3.
10 End.
```

Fixed station simply generates the messages and when the checkpoint request message is generated it signals MSSS to take checkpoint of all the hosts. On receiving this signal, MSS sends checkpoint request to all hosts. Then every host giving their checkpoint status to MSS and then MSStransfers to the fixed station and it make it to permanent. Because checkpoints are taken synchronously there is always a consistent state.

Algorithm executed at a Fixed Station:

```
1.    Begin
2.    Initialise the network variables like self and neighbours id.
3.    Create a queue.
4.    Generate the message.
```

```
5.    Check (message _ type)

6.    If (message _ type = = Normal)
      {     Execute normal operation.     }

7.    If (message _ type = = Checkpoint Request)
      {          Checkpoint Request generated.
            Send signal to MSS, for taking Checkpoint ofall the hosts.
       Wait,
             Receive acknowledgement from MSS.
      Collect information of all the hosts.
              Make them permanent. }

8.    If (message _type = = Failure Signal)
      {       Send Failure signal to all hosts and MSS.
              Suspend operation.}

9.    If (message _ type = = Recovery Signal)
              {    Sends recovery signal to MSS and all hosts.
              Wait,
      Collect all information for restarting by MSS.
              Restart from the last global consistent point.     }

10.   Goto step 4

11.   End
```

PROPOSED MODIFICATIONS

We propose to verify certain conditions after reception of message instead of taking checkpoints after fixed time interval. This helps in reducing the recovery time.

When it is time for a host to take a basic checkpoint, it takes a basic checkpoint only if it did not already take a forced checkpoint with the sequence number that is expected to be assigning to the next basic checkpoint; otherwise, it skips taking the basic checkpoint. When a host receives a message, it takes a forced checkpoint before processing the message if the sequence number of its current checkpoint is less than the checkpoint sequence number received with message.

Each mobile host has two variables seq_noi and nexti .The value of the variable seq_no denotes the sequence number of the latest checkpoint taken by mobile host. The value of next denotes the sequence number to be assigned to the next basic checkpoint that mobile host will take. The variable next is incremented by host after every checkpoint interval. The sequence number piggybacked with the message is denoted by Message. seq_no. The temporary storage of sequence

number is denoted by temp.seq_no. Since the protocol requires the control information to be piggybacked, while sending a message. The check pointing decision is taken while receiving a message.
Algorithm executed at a Mobile Host:

- Mobile host increment nexti periodically after every checkpoint interval

  nexti = nexti + 1;
- When it is time for mobile host to take basic checkpoint
  1. If nexti > seq_noi
  2. Execute procedure checkpoint.
  3. Temp.seq_no = nexti.
  4. Seq_noi = temp. seq_no.
- While mobile host sending a message
  1. Message. Seq_no=swq_noi.

     Sequence number of the current checkpoint piggybacked with message
  2. Send it to particular mobile host.
- While mobile host Mj receives a message from Mi
  1. If Message .seq_no > seq_noj
  2. Execute procedure checkpoint.
  3. temp.seq_*no= Message. seq_* no.
  4. seq_*noj = temp. seq_*no
  5. Process the message.

CONCLUSION

In the existing scheme the decision of check pointing is taken after a fixed interval of time when there is a checkpoint request message generated. In this case, if we reduce the checkpoint interval then the number of checkpoints is increased but the rollback distance is decreased. The overall effect of time needed to process the messages is increased is increased with the increase of number of messages and mobile hosts because the reduction of rollback time is counterbalanced by the overhead of checkpoint time. We propose the modification in which we verify certain condition after receiving the message.

As the check pointing decision is selective and hence, the total number of checkpoints is reduced. Again the algorithm has no additional control message overhead. So total time to process the message is less as compared to existing protocol where the checkpoint overhead is high.

REFERENCES

[1] *Birman, K., and T. Joseph," Reliable Communications in the Presence of Failures," ACM Transaction on Computer System, vol. 5, no. 1, Feb. 1989, pp. 47-76.*

[2] *D. Manivannan, M. Singhal, Asynchronous recovery without using vector timestamps, J. Parallel Distrib. Comput. 62 (12) (2002) 1695–1728.*

[3] *R. Prakash, M. Singhal, Maximal global snapshot with concurrent initiators, in: Proceedings of the Sixth IEEE Symposium on Parallel and DistributedProcessing, October 1994, pp. 334–351.*

[4] *M. Schneider, Self-stabilization, ACM Comput. Surveys 25 (1) (1993) 45–67.P.S. Mandal, K. Mukhopadhyaya / J. Parallel Distrib. Comput. 67 (2007) 816 – 829 829*

[5] *A.P. Sistla, J. Welch, Efficient distributed recovery using message logging, in: Proceedings of the ACM Symposium on Principles of Distributed Computing, 1989,pp. 223–238.*

[6] *M. Spezialetti, P. Kearns, Efficient distributed snapshots, in: Proceedings of the Sixth International Conference on Distributed Computing Systems, 1986, pp.382–388.*

[7] *R.E. Strom, S. Yemini, Optimistic recovery in distributed systems, ACM Trans. Comput. Syst. 3 (3) (1985) 204–226.*

[8] *N.H. Vaidya, Consistent logical check pointing, Technical Report 94-051,Department of Computer Science, Texas A&M University, July, 1994.*

[9] *N.H. Vaidya, Staggered consistent check pointing, IEEE Trans. Parallel Distrib. Syst.10 (7) (1999) 694–702.*

[10] *Y.M. Wang, Y. Huang, W.K. Fuchs, Progressive retry for software error recovery in distributed systems, in: 23rd Annual International Symposium on Fault-Tolerant Computing, June 1993, pp. 138–144.*

[11] *Y.M. Wang, A. Lowry, W.K. Fuchs, Consistent global checkpoints based on direct dependency tracking, Inform. Process. Lett. 50 (4) (1994) 223–230.*

[12] *Wayne Goddard, Stephen T Hededtniemi ,David P. Jacobs,Pradip K. Srimani, Zhenyu Xu, "Self Stabilizing graph protocol ",School of Computing, Clemson University Clemson SC 29634, USA*

[13] *Surender Kumar ,R.K. Chauhan and Parveen Kumar " Design And Performance Analysis Of Coordinated Check pointing Algorithms For Distributed Mobile Systems" I J D & Parallel systems (IJDPS) Vol.1, No.1, September 2010*

Authors would like to extend their heartfelt thanks to the academic and infrastructural support received from the University.