Low Overhead Handoff Based Secure Checkpointing for Mobile Hosts

Priyanka Dey and Suparna Biswas

Department of Computer Science & Engineering
West Bengal University of Technology
priyankadey24@yahoo.co.in,
mailtosuparna@gmail.com

Abstract. An efficient fault tolerant algorithm based on movement-based secure checkpointing and logging for mobile computing system is proposed here. The recovery scheme proposed here combines independent checkpointing and message logging. Here we consider mobility rate of the user in checkpointing so that mobile host can manage recovery information such as checkpoints and logs properly so that a mobile host takes less recovery time after failure. Mobile hosts save checkpoints when number of hand-off exceeds a predefined hand-off threshold value. Current approaches save logs in base station. But this approach maximizes recovery time if message passing frequency is large. If a mobile host saves log in its own memory, recovery cost will be less because log retrieval time will be small after failure. But there is a probability of memory crash of a mobile host. In that case logs can not be retrieved if it is saved only in mobile node. Hence in this algorithm mobile hosts also save log in own memory and base station. In case of crash recovery, log will be retrieved from base station and in case of transient failure recovery logs will be retrieved from mobile host. In this algorithm recovery probability is optimized and total recovery time is reduced in comparison to existing works. Logs are very small in size. Hence saving logs in mobile hosts does not cause much memory overhead. This algorithm describes a secure checkpointing technique as a method for providing fault tolerance while preventing information leakage through the checkpoint data.

Keywords: Fault-Tolerance, Mobile Computing, Checkpointing, Logging, hand-off, recovery time, crash failure, transient failure etc.

1 Introduction

Fault tolerant mobile computing systems are increasingly being used in such application as e-commerce, banking, different mobile monitoring devices in hospital and mission critical application, where privacy and integrity of data as important as uninterrupted operation of services provided.

The checkpointing and logging technique is one such distributed service to provide fault tolerance for the system. Checkpoint which is a consistent snapshot of the system contains process's state, register, segment and actual data of process [1]. Many checkpointing-recovery schemes have been proposed for the distributed systems.

N. Meghanathan et al. (Eds.): Advances in Computing & Inform. Technology, AISC 176, pp. 103–112.
springerlink.com

However, these schemes cannot be directly used in the mobile environment because of mobile computing system has many constraints [2][3] e.g. mobility, low bandwidth, less stable storage and frequent disconnection.

In light of the above constraint, this paper presents a movement-based checkpointing strategy combined with logging for recovery of individual hosts in mobile computing environments. In this approach Mobile host takes a checkpoint when handoff exceeds threshold of mobility. Our proposed approach mobile host saves log in its own memory to reduce the recovery time after failure. This type of log saving schemes is applicable to such kind of mobile device which can support large amount of storage of message and in which instant recovery is required after failure. This log saving approach is applicable in hospital while several mobile monitoring devices are attached to patient to monitor their temperature, pulse and so on. If failure is occur these devices should be recovered instantly [4]. But due to probability of memory crash, mobile hosts save log in both own memory and base station. This case is suitable in a mission critical application providing communications and shared situational awareness to an active military unit where quick and perfect recovery is required.

The network is one of the common places where security threats exist [5][6]. In this algorithm each mobile host encrypts checkpoint and saves in base station. This prevents information leakage from checkpoint while being transferred through wireless channel.

2 Related Works

An overview of different types of checkpointing and logging techniques and roll back recovery techniques based on the different types of checkpoint based and log based can be found in [7] [8].Prakash and Singhal describe in [9] a checkpointing algorithm for Mobile Computing System. This checkpoint collection algorithm is synchronous and non-blocking. A minimum number of nodes are forced to take checkpoints. Each MH maintains a dependence vector .MHs.T.Park et.al has presented an efficient movement based recovery scheme in [10]. Main feature of this algorithm is that a host carrying its information to the nearby MSS can recover instantly in case of a failure. An MH moving inside a range, recovery information remains in host MSS otherwise it moves recovery information to nearby MSS.In [11] Sapna E. George et .al describes a movement based checkpointing and logging scheme based on mobility of mobile hosts. A checkpoint is saved in mobile support station when hand-off count exceeds a predefined optimum threshold. Recovery cost is minimized in this scheme by computing optimal movement threshold.

2.1 Our Observations

In [3] checkpoint is taken on periodic basis. But this checkpoint may not be suitable for mobile environment due to following reason-(a) if the frequency of check pointing is high, the additional overhead is large. (b)If the frequency is low, the recovery cost may be very large. In [11] log is saved in BS. So recovery time increases due to log retrieval cost.

2.2 Problem Definition

Based on the above analysis we propose an efficient fault tolerant algorithm based on movement-based secure checkpointing and logging which identifies problems and tries to provide proper solutions.

(i) **Transient failure of mobile hosts:** Occurrence of transient fault can not affect memory content of mobile host. No need to save log in base station. Log can be saved and retrieved from memory of mobile host itself.

(ii) **Crash recovery:** To reduce recovery overhead we keep log in mobile host's own memory. If mobile host's memory is crashed, log saved inside memory can't recover. So we consider another case in log saving techniques where log will be saved in both base station and mobile host's memory. If memory crash occur log can be retrieved from base station.

(iii) **Security attack to checkpoint in wireless channel:** To provide fault tolerance secure check pointing is an important issue. Checkpoint is transferred to base station when it is saved and transferred from base station when mobile host is failed over insecure wireless channel. The checkpoint can be attacked, compromised and corrupted in-transit by any intruder or malicious node. The mobile can not recover using modified checkpoint data. Hence rollback recovery of a failed mobile host from last saved state will not be possible. So, security of checkpoint data in wireless network is required to ensure fault tolerance of the processes running on the mobile hosts. Here we encrypt checkpoint so that checkpoint data does not changed.

(iv) **Random movement and handoff of mobile hosts:** Mobility of mobile host is an important concern in case of mobile environment because depending on mobility, logs and checkpoints of a mobile host are saved in different base stations. So, recovery cost depends on checkpoint and log retrieval cost. The optimum movement threshold value ensures that checkpoints can be saved nearer to recovery base station, logs are not scattered too much so that overhead of unnecessary checkpoints and logs can be avoided.

(v) **Overhead:** Overhead is optimized by saving log in mobile host's own memory so that log retrieval cost will be zero during recovery from transient failure.

3 System Model

The mobile computing system considered here consists of n number mobile host (MH) and m number of base station (BS). MHs are connected to BSs through wireless network and BSs are connected with each other though wired network. An MH can communicate with other MHs and BSs only through the BS to which it is directly connected. In this system the channel between an MH is connected to a BS also ensures FIFO communication in both the directions. Proposed algorithm is non-blocking while taking checkpoint.

3.1 Assumption

- During checkpoint interval messages sent, received are saved into log file.
- Every MH takes and transfers encrypted checkpoint.

4 Data Structure and Notation

$BS_{current}$= currently connected base station id. BS_{prev}=previous visited base station id.h_c= handoff counter which keeps the record of number of handoff occurs. $BS_{log=}$base station id where logs are saved. $BS_{chkp=}$base station id where checkpoint is saved. hc = number of hopcount.lc = count the number of log. chkpt_intv=checkpoint interval. r = Ratio of bandwidth of wireless network to wired network. $T_{chkp\_take\_time}$ = time required to take checkpoint. $T_{chkpencrypt\_time}$ = time required to encrypt checkpoint. T_{rec}= time required to recovery after failures. $T_{save\_chkp\_bs}$ = time required to transfer checkpoint.T1=time required to send the checkpoint request to BSchkp. T2=time required to transfer checkpoint from BS_{chkp} to $BS_{current.}$T3= time required to save checkpoint. T4=time required to send the request of BS to send log file. T5 = time required to transfer log from BS_{log} to $BS_{current}$ .T6=time required to transfer log to MH from $BS_{current.}$ T7 = time required to replay log. T8= time required to decrypt checkpoint.

5 Proposed Work

Every time MH will connect with any BS after successful authentication. There is a handoff counter which will increment every time when MH will move one BS to another. When MH's handoff counter will exceed predefine threshold value MH will take checkpoint. During checkpointing, MH will first save the states and encrypt it and then save it in the stable storage .Then the handoff counter will initialized to zero.MH can communicate with other MH through message. The receiver MH will save the received message in log file. When failure is occurred MH will connect to any BS (not necessary the BS where failure is occurred). MH then gives the BS id to the $BS_{current}$ to get the checkpoint. The $BS_{current}$ will then send request to the BS_{chkp} where checkpoint is saved to get checkpoint. BS_{chkp} then response the request and send the checkpoint. The current BS then sends the encrypted checkpoint to the MH. MH then decrypts the checkpoint and rollback to its last taking checkpoint. In case of logging we consider two cases based on log saving techniques of different mobile computing devices. Two cases are illustrated in the following-

Case 1: MH will save log in its own memory. It will not copy the log in BS.

Case 2: MH will save log in its own memory and also copy the log in BS_{log}. In case1 there is a probability of MH memory crash. So if memory crash occurs MH can retrieve log from BS_{log} otherwise from memory.

5.1 Working Example

The above proposed work is illustrated through the following example cellular network. For example we consider 3 number of MH in this figure is. Here $C_{1,1}$ denotes

MH_1 takes its checkpoint at interval 1. When MH1 moves from one BS to the other BS its handoff counter i.e. h_c is incremented by 1.

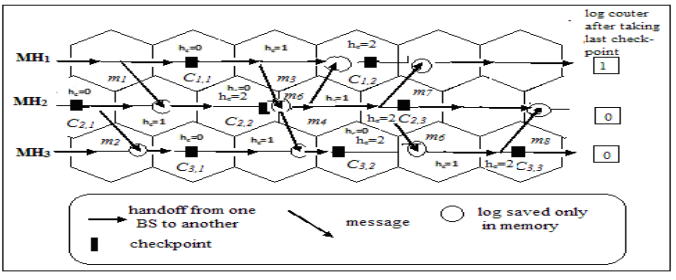

Fig. 1. Working example of our proposed algorithm

MH_1 will take checkpoint when handoff counter i.e. h_c=2.In this example we consider threshold value 2.When MH_1, MH_2, MH_3 will receive message they will first save in memory and then save to BS. Every MH also maintain log counter which count number of message received till last checkpoint. For MH_1 after taking $C_{1,2}$ it receives m_7 so log counter will be 1.After taking checkpoint log counter will be initialized to zero. Suppose MH_3 fails before taking checkpoint $C_{3,3}$ then MH_3 will rollback to $C_{3,2}$ and replay one log i.e. m_6 as its log counter will be 1 .Thus MH_3 will recover from failure.

5.2 Algorithm

1. MH initially connects to BS after successful authentication.
2. MH sends message to the another MHs. Receiver MH saves the messages in log. Then for case1 MH can save log in memory and for case2 in BS_{log}.
3. MH moves one BS to another BS and increment handoff counter.
4. If value of handoff counter is greater than movement threshold Checkpoint will be taken.
5. MH recovers after failure as explained in proposed work section:

6 Correctness and Proof

Theorem 1: The Proposed algorithm ensures consistent global checkpointing

Lemma 1: No orphan or lost message is generated by the technique

Proof: To prove this we consider two cases on the basis of Fig.1.

Case 1: MH_1 fails after taking checkpoint $C_{1,2}$ and receiving m_7 from MH_2.
In this case m_7 is considered as lost message because the state of MH_2 reflects sending it but MH_1 does not reflects receive it. So m_7 can be considered as loss message because it can't replay after failure. According to our algorithm m_7 is not considered as lost message because when MH_1 receives m_7 it saves in its own memory. After failure MH_1 rollbacks to $C_{1,2}$ and replays m_7.

Case 2: MH_2 fails before taking checkpoint $C_{2,3}$ after sending m_6 to MH_1
In this case m_6 considers as orphan message because the state of MH_3 saves m_6 is received from MH_2 but as MH_2 fails , its state has no record about the event that MH_2 sends m_6 to MH_3. So m_6 can be considered as orphan message. In our algorithm m_6 is not considered as orphan message because when MH_1 receives m_6 it saves in its own memory and can replay after failure. So theorem1 is proved.

7 Performance Analysis

We simulate this algorithm for mobile environment using C language to get different parameters for the performance analysis of the proposed algorithm.. We simulated encryption and decryption of checkpoints using a very simple cryptography technique to verify the working of the proposed secure checkpointing algorithm. In Practical purpose strength of security technique is a big issue. Public key cryptography using elliptic curve cryptography (ECC) is already well established for PDAs. Implementation of ECC algorithm is out of scope of this work. Only encryption and decryption times of checkpoint are taken same as ECAES encrypt and ECAES decrypt time mentioned in [12]. In our system the following parameter values are kept constant. The ratio of bandwidth of wireless to wired network is .1 [11]. Size of each character of each message is considered as 1 bit. Failure rate and log arrival rate are considered to be of exponential distribution. These values do not assume a specific application or environment and are chosen for simulation and performance analysis only.

In this performance analysis we compare the result of log taking scheme used in [11] with our log taking scheme. For this comparison we consider three cases. We consider this to compare with our own case to calculate the overhead. These cases are:

Case 1: MH will save the log to the BS where it will save checkpoint and delete the message from own memory. MH will retrieve log from BS_{log} for recovery.

Case 2: MH will save log in its own memory. It will not copy the log in BS.

Case 3: MH will save log in its own memory and also copy the log in BS_{log}.In case memory is crashed it can retrieve log from BS_{log} otherwise from MH's memory.

- **Secure Checkpoint Cost**

This cost will be same for case1, case2 and case3 because checkpoint taking techniques for three cases are same.

$$T_{chkp\_take\_time} + T_{chkpencrypt\_time+} T_{save\_chkp\_bs}$$
$$= .003+.234+1.759 = 1.996s$$

If we consider encrypted checkpoint overhead will increase by 1.759s.

- **Recovery Time**

For case 1: $T_{rec=}(T1_{+}hc*(T2))*r+T3_{+}T4_{+}lc_{*}hc*(T5)*r+T6_{+(\ lc\ *\ T7)}$
For case 2: $T_{rec\ =}(T1\ _{+}hc*(T2))*r+\ T3\ _{+}\ T8_{+}\ T1\ _{+(\ lc\ *\ T7)}$
For case 3: $T_{rec\ =}(T1\ _{+}\ hc\ *(\ T2)\ )*r\ +T3\ _{+}\ T8_{+}p*_{(}lc_{*}T7_{)})+((1\text{-}p)*(T4_{+}lc_{*}hc*(\ T5)*r$
$+\ \ T6_{+(\ lc\ *\ T7))}$

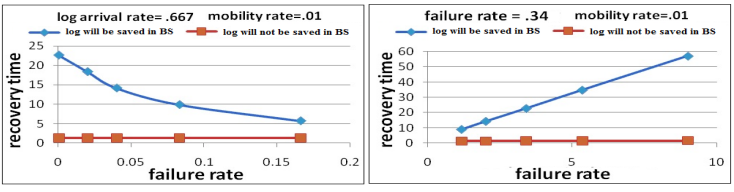

Fig. 2. (a) Failure vs. recovery time log. (b) log arrival rate vs. recovery time.

In fig.2. (a) we take log arrival rate .667 as constant. By varying failure rate we get recovery time we see that if failure rate increases recovery time will decreases because if failure rate increases less log will be taken. In this figure we compare the result of case1 and case2. In Fig.2.(b).We get recovery time by varying log arrival rate and keep failure rate .34 as constant. We see that if log arrival rate increases recovery time will increase because if log arrival rate is increased less number of logs will be taken. So recovery time will be less. In this figure we compare the result of case1 and case2. In both figure we see that recovery time overhead increases in case1 than case2 as in case2 log arrival cost does not consider.

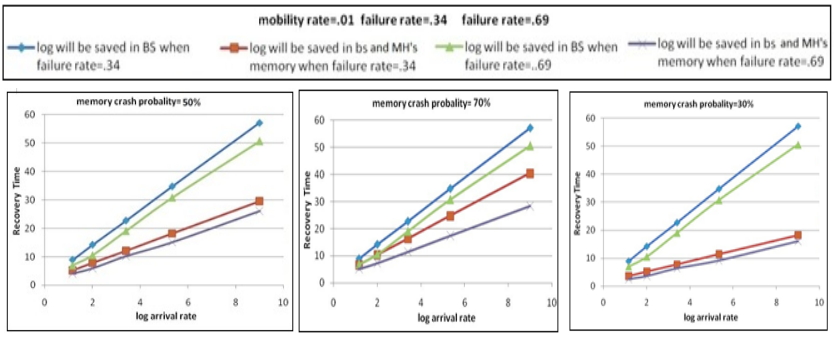

Fig. 3. Log arrival rate vs. recovery time (a) when memory crash probability=50% (b) when memory crash probability=70% (c) when memory crash probability=30%

In the above all the figure we compare the recovery time of case1 with case3 by varying log arrival rate and keeping constant the failure rate .34 and .69.In Fig.3 (a) according to our result the recovery time overhead for case1 is increases 13.006s than

case3 for failure rate=.34 and 6.8257s than case3 for failure rate=.69. In Fig.3 (b) 7.832s for failure rate=.34 and 5.2794s for failure rate=.69.In Fig.3 (c) 18.27s for failure rate=.34 and 11.774s for failure rate=.69.

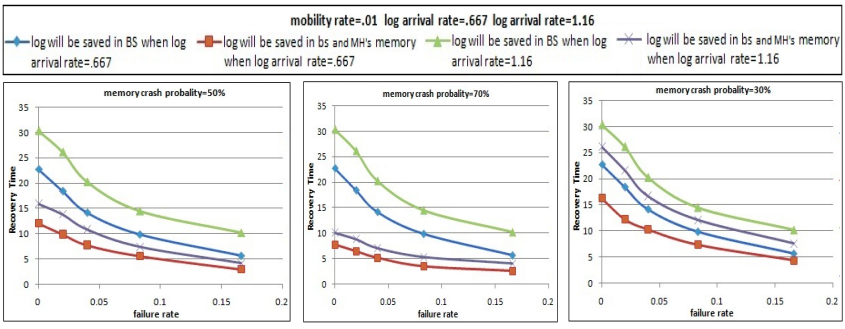

Fig. 4. Failure rate vs.recovery time (a) when memory crash probability=50% (b) when memory crash probability=70% (c) when memory crash probability=30%

In all the above figure we compare the recovery time of case1 with case3 time by varying failure rate and keeping constant the log arrival rate.667 and 1.16.In Fig.4(a) we see that the recovery time overhead for case1 increases 7.6625s than case3 for log arrival rate=.667 and 9.8685s for log arrival rate=1.16.In the Fig.4(b) 4.for log arrival rate=.667 and 5.6672s for log arrival rate=1.16.In the Fig.4(c) 8.77s for log arrival rate=.667 and 13.221s for log arrival rate=1.16.

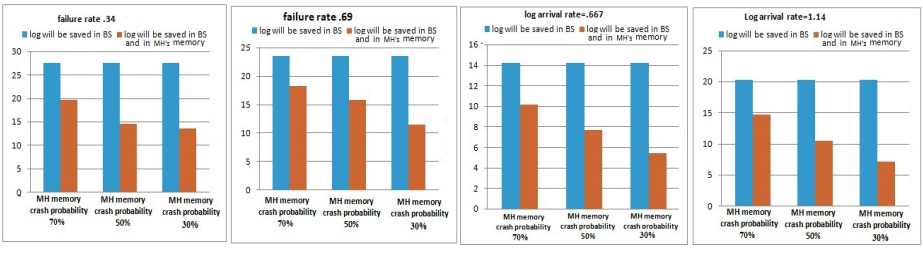

Fig. 5. Recovery time overhead comparison between case 1 and case 3 for memory crash probability 30%, 50%, & 70% (a) for failure rate .34 and .69. (b) log arrival rate .667 and 1.16.

In the above Fig.5.(a) and 5.(b) as memory crash probability increases we can see that recovery time overhead decreases because probability of retrieve log from BS increases with memory crash probability. So we can see that case3 also gives less recovery time than case1.We encrypts checkpoint in our proposed algorithm. In the below we analyze for encryption how much time is increased.

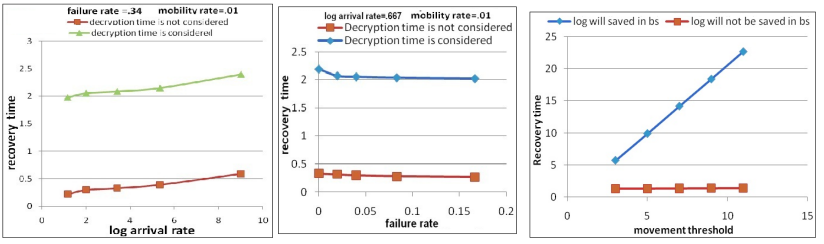

Fig. 6. (a) Log arrival rate vs recovery time and failure rate vs recovery time. (b) Recovery time vs. movement threshold.

In the above Fig.6.(a) we calculate recovery time by considering decryption time of checkpoint.and not considering decryption.We can see that recovery time overhead increases by 1.759s.In Fig.6.(b) we compare recovery time of case1 with case2 by varying movement threshold. We can see if movement threshold increases recovery time also increases because if movement threshold increases checkpoint interval between two checkpoint increases so checkpoint transfer cost will be high and more number will be taken between this interval.Recovery time for case1 is more than case2 because in case2 log is saved in own memory.

8 Conclusions

Mobile computing has been developing very rapidly in recent years. Some of the checkpointing and recovery techniques proposed for mobile computing systems did not take checkpoints regard to the mobility rate of the user and unnecessarily incur additional overhead in maintaining recovery data. In our proposed approach in case of transient failure logs are retrieved from mobile host's own memory to reduce the recovery time after failure. But due to probability of memory crash, mobile hosts save log both in their own memory and base station. In case of crash failure logs are retrieved from base station. Saving two copies of log may seem to cause memory overhead but log searching and transfer cost from base station gets reduced. This algorithm proposes a secure checkpointing system as a method for providing checkpointing capability while simultaneously preventing information leakage of application data saved in checkpoint.

References

1. Gman, H., Ahn, S.J., Han, S.C., Park, T., Yeom, H.Y., Cho, Y.: Kckpt: Checkpoint and Recovery Facility on UnixWare Kernel. Computers and Applications, 303–308 (2000)
2. Forman, G.H., Zahorjan, J.: Challenges of Mobile Computing. Journal Computer 27(4), 31–40 (1994)
3. Schiller, J.: Mobile Communications, 2nd edn. Addison Wesley
4. Adis, W.: Mobile Computing for Hospitals: Transition Problems. Communications IIMA 5(2), 67–76 (2005)
5. Nam, H., Kim, J., Hong, S.J., Lee, S.: Secure checkpointing. Journal of Systems Architecture 48, 237–254 (2003)

6. Agrawal, D.P., Deng, H., Poosarla, R., Sanyal, S.: Secure Mobile Computing. In: Das, S.R., Das, S.K. (eds.) IWDC 2003. LNCS, vol. 2918, pp. 265–278. Springer, Heidelberg (2003)
7. Mootaz Elnozahy, E.N., Alvisi, L., Wang, Y.-M., Johnson, D.B.: A Survey of Rollback Recovery Protocol in Message Passing System. ACM Comput. Surv. 34(3), 375–408 (2002)
8. Kumar, P., Garg, R.: Checkpointing Based Fault Tolerance in Mobile Distributed Systems. International Journal of Research and Reviews in Computer Science (IJRRCS) 1(2), 83–93 (2010)
9. Prakash, R., Singhal, M.: Low Cost Checkpointing and Failure Recovery in Mobile Computing Systems. IEEE Transacrions on Parallel and Distrinuted Systems 7(10), 1–38 (1996)
10. Park, T., Woo, N., Yeom, H.Y.: An Efficient recovery scheme for fault-tolerant mobile computing systems. Future Generation Computer System 19(1), 37–53 (2003)
11. George, S.E., Chen, I.-R., Jin, Y.: Movement-Based Checkpointing and Logging for Recovery in Mobile Computing Systems. In: MobiDE 2006 Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 51–58 (2006)
12. Lopez, J., Dahab, R.: An overview of Elliptic Curve Cryptography, Technical Report IC-00-10. State University of Campinas, pp. 1–34 (2000)