HANDOFF BASED SECURE CHECKPOINTING AND LOG BASED ROLLBACK RECOVERY FOR MOBILE HOSTS

Priyanka Dey[1] and Suparna Biswas[2]

[1]Department of Computer Science & Engineering, West Bengal University of Technology, Salt Lake, Kolkata
priyankadey24@yahoo.co.in
[2]Department of Computer Science & Engineering, West Bengal University of Technology, Salt Lake, Kolkata
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. If the failure is transient then logs can be retrieved from the memory of 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. Hand-off threshold is a function of mobility rate, message passing frequency and failure rate of mobile hosts. 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.

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. 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 (MH) takes a checkpoint when handoff exceeds threshold of mobility. The threshold depends on log arrival rate, failure rate and mobility rate of MH. Our proposed approach mobile host saves log in its own memory to reduce the recovery time and increase recovery probability 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. We prove here our log saving technique takes less time and more recovery probability than already proposed log saving technique through our simulation results. This case is suitable in a mission critical application providing communications and shared situational awareness to an active military unit. The failure rate and mobility rate in such a mobile environment is likely to be very high. At the same time, fast recovery from failures is more important than minimizing cost of failure-free operations. For this type of system after failure it is necessary to retrieve every message and instant recovery is preferable.

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. Similarly at the time of recovery, encrypted checkpoint will be retrieved and decrypted by the failed mobile host. Thus our proposed low overhead, movement based, secure checkpointing and rollback recovery algorithm ensures fault tolerance of applications running on mobile hosts as well as confidentiality of checkpoint data.

The rest of the paper is structured as follows: Section 2 discusses some of the related works, our observation and problem definitions. Section 3 describes system model & preliminary assumptions. Section 4 elaborates data structures and notations used. Section 5 explains proposed checkpointing scheme, basic ideas and describes the algorithm. Section 6 gives Necessary correctness proofs. Section 7 elaborates simulation and performance analysis. In Section 8 we conclude our work.

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 mobile host (MH)maintains a dependence vector of 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 mobile support station (MSS) can recover instantly in case of a failure. For failure-free execution there is a concept of a ‘certain range’ is introduced. An MH moving inside a range , recovery information remains in host MSS otherwise it moves recovery information to nearby MSS.

Sapna E. George et .al [11] describes a checkpointing and logging scheme based on mobility of MHs. A checkpoint is saved when hand-off count exceeds a predefined optimum threshold. Recovery probability is calculated and recovery cost is minimized in this scheme. In case of logging when MH receives message, saves this log in MSS. They compute recovery time and recovery probability by varying log arrival rate and failure rate.

2.1. Our Observations

Many checkpointing-recovery schemes have been proposed for the distributed systems; however, these schemes cannot be directly used in the mobile environment because of mobile computing constraints. In [10] and [11] it is described that independent checkpointing and message logging is suitable for failure recovery in mobile devices if we consider movement based checkpointing and logging.

In [10] they will take checkpoint on periodic basis. Periodic checkpointing may not be suitable for mobile environment due to following reason-

1. If the frequency of check pointing is high, the additional overhead is large.
2. If the frequency is low, the recovery cost may be very large.

In [11] a movement based checkpointing and logging scheme is introduced in which checkpoint is taken based on movement threshold instead of periodic. In this algorithm log is saved in base station. So recovery time increases due to log retrieval cost.

Security is also an important issue in case of wireless network. In [10] and [11] checkpoint is always saved in base station. But after failure Checkpoint needs to be transferred. So checkpoint data may change where it is saved or when it is transferred. So security of checkpoint is required.

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.

- **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.
- **Crash recovery**: If we save log only in base station recovery overhead will increase. So we consider a case in log saving techniques where log will be kept 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.

- **Security attack to checkpoint in wireless channel**: Mobile computing system is used in various financial transactions where fault tolerance is necessary. To provide fault tolerance secure check pointing is an important issue. Checkpoint is transferred to base station over insecure wireless channel. Again if a mobile host fails, checkpoint needs to be transferred to the mobile host over 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.
- **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.
- **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 of mobile nodes called mobile host (MH) and m number of static nodes called 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 through messages. An MH can be directly connected to at most one BS at any given time. An MH can communicate with other MHs and BSs only through the BS to which it is directly connected. The links in the wired network support FIFO message communication. In this system the channel between an MH is connected to a BS also ensures FIFO communication in both the directions. Message transmission through these links takes an unpredictable, but finite amount of time during transmission. During normal operation, no messages are lost in transit. The system does not have any shared memory or a global clock among nodes. So, all communication and synchronization takes place through messages. Proposed algorithm is non-blocking while taking checkpoint.

3.1. Assumptions

- During checkpoint interval messages sent, received are saved into log file.
- Every MH takes and transfers encrypted checkpoint.
- Mobile host will move only in forward direction.

4. DATA STRUCTURE AND NOTATION

mh_id = mobile host id. $BS_{current}$= 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. Movement_threshold=number of handoff required to take checkpoint. 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 BS_{chkp}. 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_{log} 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. RP=recovery probability of mobile host. p=percentage of memory crash.

5. PROPOSED WORK

The system is composed of n number of mobile nodes called MH and m number of stationary nodes called BS. When MH will connect with first BS it has to save its code through which it can be authenticated. When MH will change BS it will be authenticated first. There is a handoff counter which will increment every time when MH will move one cell to another. We consider some threshold value. When MH's handoff counter will exceed the 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. Log file will contain the message, the sender and receiver ID, message sequence no. and the interval of receiver.

When failure is occurred MH will connect to any BS (not necessary the BS where failure is occurred) .When it will connect with any BS authentication will occur. If authentication is successful MH can connected to BS. 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-

Case1: MH will save log in its own memory. It will not copy the log in BS. In this case when failure is occurred it has to only retrieve the checkpoint. Then MH will rollback to its last checkpoint and replay the log.

Case2: 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 number of MH in this figure is. Here $C_{1,1}$ denotes MH_1 takes its checkpoint at interval 1.When MH_1 will take next checkpoint the checkpoint is denoted by $C_{1,2.}$The checkpoint for other MH's are denoted in the similar way. When MH1 moves from one BS to another BS its handoff counter i.e. h_c is incremented by 1.

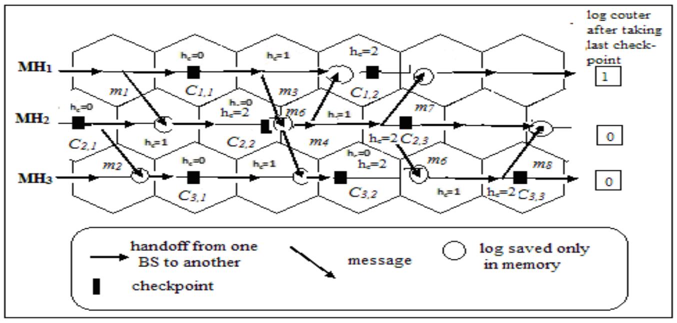

log couter after taking last check-point
MH1
MH2
MH3
handoff from one BS to another
message
log saved only in memory
checkpoint

Figure 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.So When h_c=2 every MH in this example will take checkpoint. No process has to block checkpoint. No process has to communicate with each while taking checkpoint. When MH1 will take point h_c will initialized to zero and h_c will be again incremented when MH_1will move to another BS. This will be applicable for MH_2 and MH_3.When MH_1, MH_2, MH_3 will receive message they will first save in memory and then save to BS. So if memory crash is occurred MH can retrieve log from 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. If failure is occurred before taking checkpoint MH check log counter to decide how many log it has to replay. 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. After encryption checkpoint will be saved in $BS_{chkp.}$
5. After failure recovery process of MH is explained in the following pseudo code.

```
 failure Recovery()
{
  failed MH reconnects to any BS arbitratrily ;
  MH will give the $BS_{chkp}$ id where checkpoint is saved .
  $BS_{current}$ will send request to the $BS_{chkp}$ where check
  point is saved .
  $BS_{chkp}$ then send checkpoint .
  $BS_{current}$ then send the encrypted checkpoint to MH.
  MH then decrypt checkpoint.

if(failure is memory crash){
  MH will give the $BS_{log}$ id where logs are saved.
  $BS_{current}$ will send request to the $BS_{log.}$
  $BS_{log}$ then send log to $BS_{current}$.
  $BS_{current}$ then send the logs to MH.
}
else
{
 MH will retrieve all the log from memory.
}
MH will rollback to last checkpoint and replay all the log.
}
```

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.-

Case1: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 cant 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 when MH_1 rollbacks to $C_{1,2}$ and replays m_7.

Case2: 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.

According to our algorithm m_6 is not considered as orphan message because when MH_1receives m_6 it saves in its own memory. So after failure MH_1 can replay it.

So theorem1 is proved.

7. PERFORMANCE ANALYSIS

We simulate this algorithm for mobile environment using C language. We use rand () function in C to generate join, leave, send and receive function randomly. 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 is already well established for PDAs. Implementation of Elliptic curve cryptography algorithm is out of scope of this work. We have implemented random movement, handoff, computation message sending and receiving, logging, checkpoint, failure and rollback recovery of mobile hosts to get different parameters for the performance analysis of the proposed algorithm. 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 as mentioned in [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. In this system many parameters such log arrival rate, failure rate, and movement threshold were varied across runs. These values do not assume a specific application or environment. These values were chosen for simulation and performance analysis only. Here we consider MH will move only in forward direction. Here we calculate failure rate by assuming failure is occurred when handoff counter=movement threshold-1.

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

Case1: 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.

Case2: MH will save log in its own memory. It will not copy the log in BS_{log}. This case is considered only for transient failure. But there is a possibility of memory crash. So we consider case3.

Case3: 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.

$T_{chkp\_take\_time} + T_{save\_chkp\_bs} + T_{chkpencrypt\_time}$

$=.003+.234+1.759 =1.996s$

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

- **Recovery Time**

For case1: $T_{rec} =(T1+hc*(T2))*r+T3+T4+lc*hc*(T5)*r+T6+(lc * T7)$

For case2: $T_{rec} =(T1 +hc*(T2))*r + T3 + T8+ (lc * T7)$

For case3: $T_{rec} = (T1 + hc *(T2))*r +T3 + T8+p*(lc*T7)) + ((1-p)*(T4+lc*hc*(T5)*r + T6+ (lc * T7))$

- **Recovery Probability**

For case1: RP= (No. of times successfully retrieve checkpoint and logs)/ (No. of attempts to retrieve checkpoint and log)

For case2: RP= (No. of times successfully retrieve checkpoint)/(No. of attempts to retrieve checkpoint)

For case3: RP= p*((No. of times successfully retrieve checkpoint and logs)/ (No. of attempts to retrieve checkpoint and log))+(1-p)*((No. of times successfully retrieve checkpoint)/(No. of attempts to retrieve checkpoint))

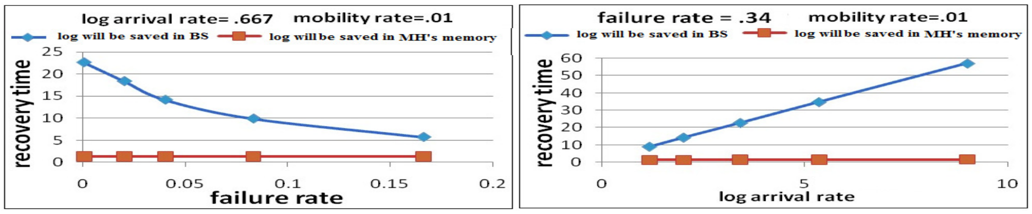


Figure 2(a).failure vs. recovery time log. (b).log arrival rate vs. recovery time

In figure 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. So during recovery less no. of log needs to be transferred. In this figure we compare the result of case1 and case2.

In figure 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 more number of logs will be taken. So recovery time will be increased. In this figure we compare the result of case1 and case2.

In both figure we can see that recovery time overhead increases in case1 than case2 as in case2 log transfer cost does not consider as log will be saved in mobile host's own memory.

By considering this memory crash probability we compute recovery time by varying log arrival rate and keeping constant the failure rate .34 and .69.

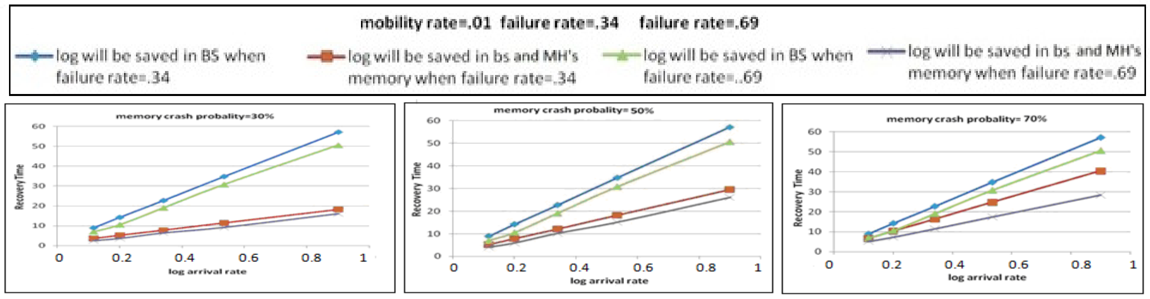


Figure 3 log arrival rate vs. recovery time (a) when memory crash probability=30%(b) when memory crash probability=50% (c) when memory crash probability=70%

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 the above figure 3(a) we compare the recovery time of case1 with case3 by considering memory crash probability =30%.According to our result the recovery time overhead for case1 increases 18.27s than case3 while considering constant failure rate=.34 and 11.774s than case3 while considering constant failure rate=.69. Recovery time of case1 is greater than case3 because only 30% time (if memory crash probability 30%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

In the above figure 3(b) we compare the recovery time of case1 with case3 by considering memory crash probability =50%.According to our result the recovery time overhead for case1 is increases 13.006s than case3 while considering constant failure rate=.34 and 6.8257s than case3 while considering constant failure rate=.69. Recovery time of case1 is greater than case3 because only 50% time (if memory crash probability 50%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

In the above figure 3(c) we compare the recovery time of case1 with case3 by considering memory crash probability =70%. According to our result the recovery time overhead for case1 increases 7.832s than case3 while considering constant failure rate=.34 and 5.2794s than case3 while considering constant failure rate=.69. Recovery time of case1 is greater than case3 because only 70% time (if memory crash probability 70%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

Now by considering this memory crash probability we compute recovery time by varying failure rate and keeping constant the log arrival rate .667 and 1.16.

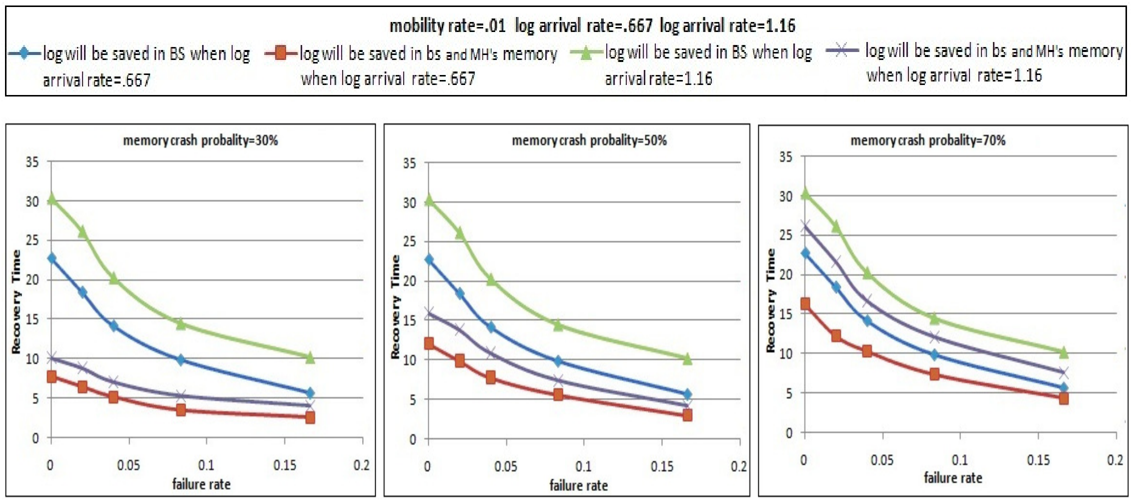


Figure 4 Failure rate vs.recovery time (a) when memory crash probability=30% (b) when memory crash probability=50% (c) when memory crash probability=70%

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 the above figure 4(a) we compare the recovery time of case1 with case3 by considering memory crash probability =30%. According to our result the recovery time overhead for case1 increases 8.77s than case3 while considering constant log arrival rate=.667 and 13.221s than case3 while considering constant log arrival rate=1.16. Recovery time of case1 is greater than case3 because only 30% time (if memory crash probability 30%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

In the above figure 4(b) we compare the recovery time of case1 with case3 by considering memory crash probability =50%.According to our result the recovery time overhead for case1 increases 7.6625s than case3 while considering constant log arrival rate =.667 and 9.8685s while considering constant log arrival rate=1.16. Recovery time of case1 is greater than case3 because only 50% time (if memory crash probability 50%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

In the above figure 4(c) we compare the recovery time of case1 with case3 by considering memory crash probability =70%.According to our result the recovery time overhead for case1 increases 4.032s than case3 while considering constant log arrival rate=.667 and 5.6672s while considering constant log arrival rate=1.16. Recovery time of case1 is greater than case3 because only 70% time (if memory crash probability 70%) log will be retrieved from base station otherwise log retrieved from own memory. So case3's recovery time is less than case1.

In the above figure if we observe the differences of recovery time between case1 and case3 by varying log arrival rate and keep constant failure rate and vice versa we can get the following figure which indicates the approx difference between case1 and case3

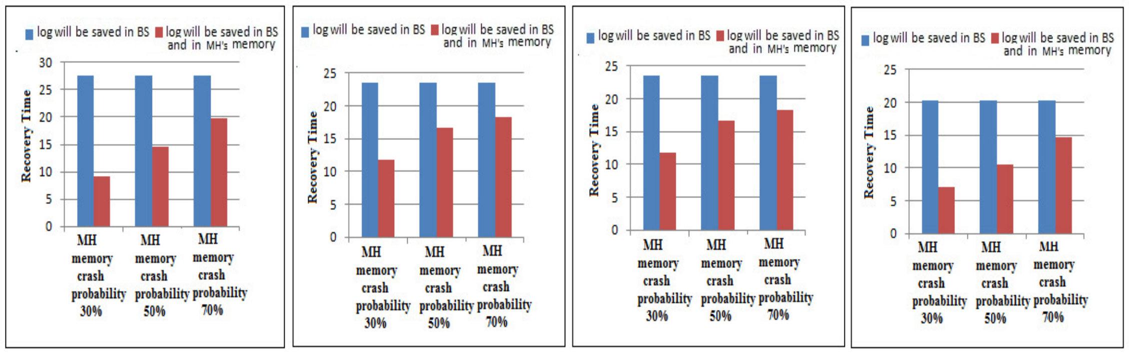


Figure.5. recovery time overhead comparison between case1 and case3 for memory crash probability 30%, 50%, & 70% (a) varying log arrival rate and constant failure rate .34(b) varying log arrival rate constant failure rate .69 (c) for varying failure rate and constant log arrival rate .667 (d) for varying failure rate and constant log arrival rate 1.16.

In the above figure 5(a) and 5(b) as memory crash probability increases we can see that recovery time overhead decreases because probability of retrieve log from base station increases with memory crash probability. So we can see that case3 also gives less recovery time than case1.This two figures are drawn after observing the recovery time differences in figure 3(a), 3(b) and 3(c).

In the above figure 5(c) and 5(d) as memory crash probability increases we can see that recovery time overhead decreases because probability of retrieve log from base station increases with memory crash probability. So we can see that case3 also gives less recovery time than case1.This two figures are drawn after observing the recovery time differences in figure 4(a), 4(b) and 4(c).

We encrypts checkpoint in our proposed algorithm. In the below we analyze for encryption how much time is increased.

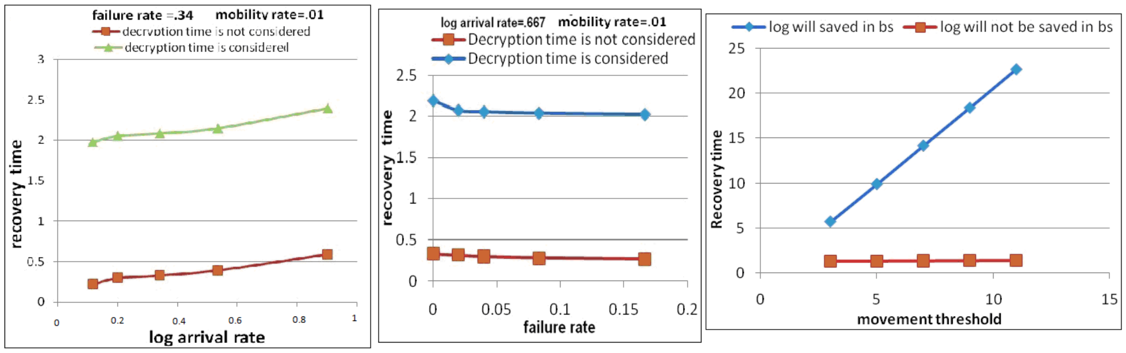


Figure 6(a) log arrival rate vs recovery time(for comparision decryption time overhead) (b) failure rate vs recovery time(for comparision decryption time overhead) (c) Recovery time vs. movement threshold

In the above figure 6(a) and 6(b) we consider two cases. In case1 we calculate recovery time by considering decryption time of checkpoint. In case2 we calculate recovery time by not considering decryption. We can see that recovery time overhead increases by 1.759s.

In figure 6(c) 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.

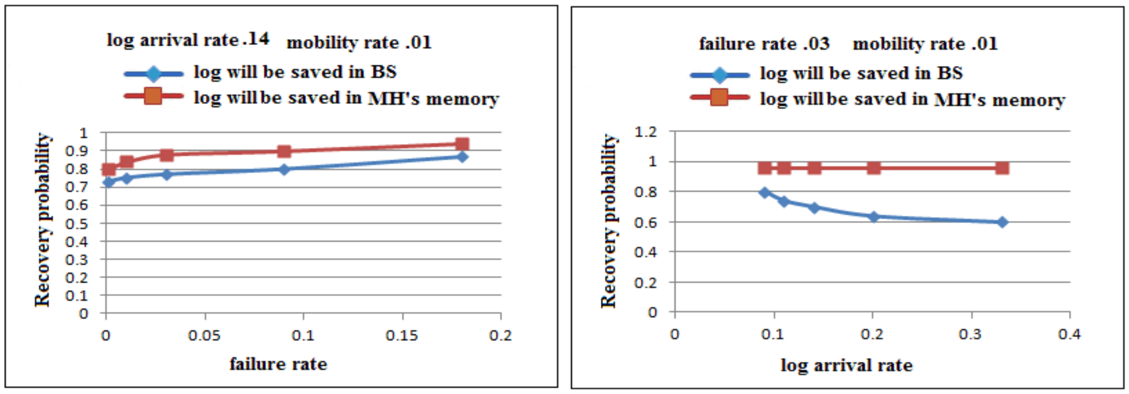


Figure 7(a) failure rate vs recovery probability (b) Log arrival rate vs recovery probability

Failure rate is directly proportional to recovery probability.If failure rate is increased rate of change of base station may decrease.So mobile host may recconect to base station which is nearest to the base station where checkpoint is saved.So probability to retrieve checkpoint is increased.If failure rate is frequent mobile host will take less no. of log.So successful log retrieval probability will also increase.Thus if failure rate is increased recovery probability will increased.In the above figure 7(a) we compare failure rate vs. recovery probability of case1 and case2.We can see that recovery probability of case2 is higher than case1 because in case1 both checkpoint and logs will be saved in base station.So after failure both checkpoint and log needs to be retrieved.But in case of case2 only checkpoint is saved in base station.So only checkpoint needs to be retrieved after failure as logs are always saved in mobile host’s memory. So after failure if logs are not retrieve properly mobile host cant recover properly.As this log retrieval does not exist in case2, recovery probability of case2 is greater than case1.

Log arrival rate is inversely proportional to recovery probability.If log arrival rate is high more no. of log needs to be retrieved. In the above figure 7(b) recovery probability of case2 is higher than case1 as in case1 logs are always saved in base station instead of mobile host's memory.So recovery probability of case2 and case1 differs as in case1 there is a chance of inproper retrieval of log for which mobile host cant recover after failure.

If we observe the abaove fiigure we can get the difference of recovery probability between case1 and case2. This difference is represented through the following graph.

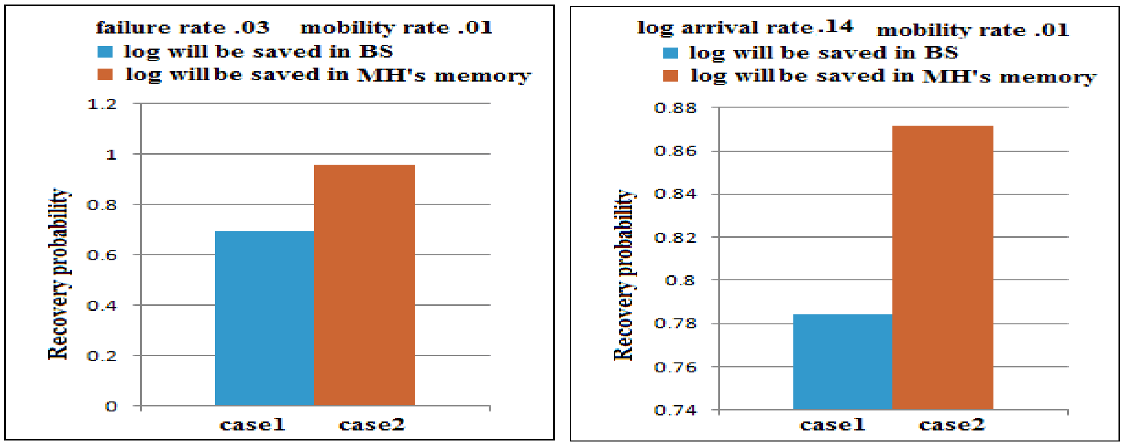


Figure.8 recovery probability overhead comparison between case1 and case2 for.(a) varying log arrival rate(b) varying failure rate

In the above figure 8(a) we can show that recovery probability of case2 is .158 higher than recovery probability of case1.Here we vary log arrival rate and keep constant failure rate as .03.After observation of figure 7(a) we draw this figure 8(a).

In the above figure 8(b) we can show that recovery probability of case2 is .06 higher than recovery probability of case1.Here we vary failure rate and keep constant log arrival rate as .14. After observation of figure 7(b) we draw this figure 8(b).

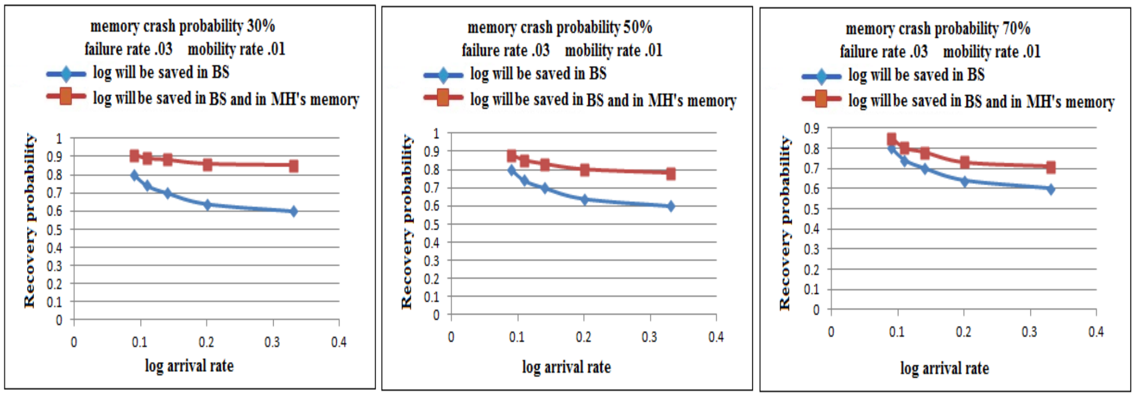


Figure.9 log arrival rate vs. recovery probability (a) when memory crash probability=30%(b) when memory crash probability=50% (c) when memory crash probability=70%

In the above figure 9(a) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 30% by varying log arrival rate and keep constant failure rate as .03. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 30%.But in case1 every time logs need to be retrieved when failure is occurred. So recovery probability of case3 is greater than case1.

In the above figure 9(b) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 50% by varying log arrival rate and keep constant failure rate as .03. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 50%.But in case1 every time logs need to be retrieved when failure is occurred. So recovery probability of case3 is greater than case1.

In the above figure 9(c) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 70% by varying log arrival rate and keep constant failure rate as .03. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 70%.But in case1 every time logs need to be retrieved when failure is occurred. So recovery probability of case3 is greater than case1.

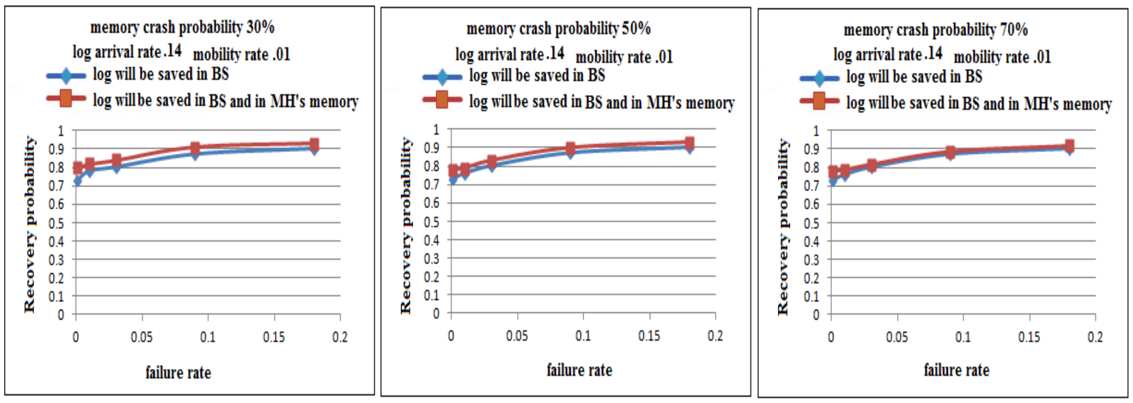


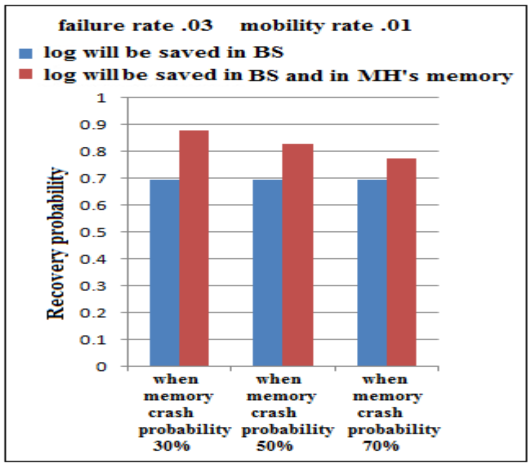
Figure.10. failure rate vs. recovery probability (a) when memory crash probability=30%(b) when memory crash probability=50% (c) when memory crash probability=70%

In the above figure 10(a) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 30% by varying failure rate and keep constant log arrival rate as .14. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 30%.But in case1 every time logs need to be retrieved when failure is occurred. Recovery probability increases with failure rate. So recovery probability of case3 is greater than case1.

In the above figure 10(b) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 50% by varying failure rate and keep constant log arrival rate as .14. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 50%.But in case1 every time logs need to be retrieved from base station when failure is occurred. Recovery probability increases with failure rate. So recovery probability of case3 is greater than case1.

In the above figure 10(c) we compare recovery probability of case1 and case3.Here we consider memory crash probability = 70% by varying failure rate and keep constant log arrival rate as .14. In this case recovery probability of case3 is higher than case1 because in case3 mobile host will retrieve logs from base station when memory crash occurs i.e. 70%.But in case1 every time logs need to be retrieved from base station when failure is occurred. Recovery probability increases with failure rate. So recovery probability of case3 is greater than case1.

If we observe the above figure we can get the difference of recovery probability between case1 and case3. This difference is represented through the following graph.



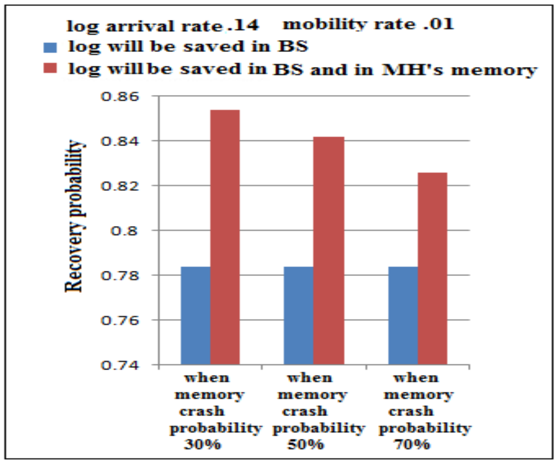


Figure.11. recovery probability overhead comparison between case1 and case3 for memory crash probability 30%, 50%, & 70% (a) for varying log arrival rate and constant failure rate .03 (b) for varying failure rate and log arrival rate .14

In the above figure 11(a) we can show that recovery probability of case3 is .182 higher than recovery probability of case1 when memory crash probability =30%.Recovery probability of case3 is .132 higher than recovery probability of case1 when memory crash probability =50%.Recovery probability of case3 is .08 higher than recovery probability of case1 when memory crash probability =70%.In all the cases we vary log arrival rate and keep constant failure rate as .03.We can see that difference between recovery probability case1 and case3 decreases as memory crash probability increases.

In the above figure 11(b) we can show that recovery probability of case3 is .042 higher than recovery probability of case1 when memory crash probability =30%.Recovery probability of case3 is .03 higher than recovery probability of case1 when memory crash probability =50%.Recovery probability of case3 is .014 higher than recovery probability of case1 when memory crash probability =70%.In all the cases we vary failure rate and keep constant log arrival rate as .14. We can see that difference between recovery probability case1 and case3 decreases as memory crash probability increases.

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 mobile host and unnecessarily incur additional overhead in maintaining recovery data. In our proposed scheme, a mobile host takes a checkpoint only after its handoff count exceeds a predefined threshold value. 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 logs 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] G.Hong, S.J.Ahn, S.C.Han, T. Park, H.Y. Yeom, Y.Cho,(2000) " Kckpt: Checkpoint and Recovery Facility on UnixWare Kernel" [J], Computers and Applications, pp. 303-308.

[2] G.H.Forman, J.Zahorjan, (1994)" Challenges of Mobile Computing ", Journal Computer Volume 27 Issue 4, April IEEE Computer Society Press Los Alamitos, CA, USA. pp. 31-40.

[3] Mobile Communications (2nd Edition), by Jochen Schiller, Publisher: Addison Wesley.

[4] W.Adis,(2005)"Mobile Computing for Hospitals: Transition Problems", Communications IIMA 67 volume 5 Issue 2. pp. 67-76.

[5] H.Nam , J.Kim , S.J.Hong , S.Lee, (2003) "Secure checkpointing", Journal of Systems Architecture 48,pp. 237–254.

[6] D.P.Agrawal, H.Deng, R.Poosarla, S.Sanyal, (2003) "Secure mobile computing", IWDC,– Springer.pp.1-9

[7] E.N. (Mootaz) Elnozahy, L.Alvisi, Y.Wang and D.B. Johnson, (September 2002), "A Survey of Rollback Recovery Protocol in Message Passing System"ACM Comput. Surv., Vol. 34, No. 3. pp. 375-408.

[8] P.Kumar, R.Garg,(2010)"Checkpointing Based Fault Tolerance in Mobile Distributed Syst-ems".International Journal of Research and Reviews in Computer Science (IJRRCS), Vol. 1, No. No. 2, June,pp83-93.

[9] R.Prakash, M.Singhal, (1996) "Low Cost Checkpointing and Failure Recovery in Mobile Computing Systems", IEEE Transacrions on Parallel and Distrinuted Systems, vol. 7, october .pp.1-38.

[10] T.Park, N. Woo, H.Y. Yeom,(2003) "An Efficient recovery scheme for fault-tolerant mobile computing systems", Future Generation Computer System, pp. 37-53.

[11] S.E.George,I.R.Chen,Y.Jin,(2006)"Movement-Based Checkpointing andLogging for Recovery in Mobile Computing Systems", MobiDE '06 Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access ,pp. 51-58.

[12] J.Lopez, R.Dahab,(2000) "An overview of Elliptic Curve Cryptography", Technical Report IC-00-10, State University of Campinas,.pp:1-34

Authors

Priyanka Dey: She completed her B.Tech in Information Technology from Techno India College of Technology (affiliated to West Bengal University of Technology). Presently she is pursuing her M.Tech in Software Engineering (Dept. of Computer Science & Engg.) from West Bengal University of Technology. Her research interest is "Fault tolerance in mobile computing".

Suparna Biswas: She obtained her B.Tech in Electronics & Communication Engineering from University of Kalyani and M.E. in Software Engineering from Jadavpur University. She is currently working as an Assistant Professor in the Dept. of Computer Science & Engg. in West Bengal University of Technology. Her areas of research interests are Fault Tolerant Mobile Computing, Software Engineering.