Pitfalls in Minimum-Process Coordinated Checkpointing protocols for Mobile Distributed Systems

R K Chauhan*, Parveen Kumar,**

**Deptt. of Computer Sc & Applications*
Kurukshetra University, Kurukshetra-136119 (Haryana)

***Deptt. of Computer Sc & Engineering*
National Institute of Technology, Hamirpur (HP), 177005

pk223475@yahoo.com

Abstract

Minimum-process coordinated checkpointing is a preferred approach to introduce fault tolerance in mobile distributed systems transparently. This approach is domino-free, requires at most two checkpoints of each process on stable storage, and forces only interacting processes to checkpoint. Sometimes, it also requires piggybacking of information onto normal messages, blocking of the underlying computation or taking some useless checkpoints. In this paper, we point out some complexities and inconstancies in existing minimum-process blocking coordinated checkpointing protocols for Mobile Distributed Systems. We also point out the cause of the problems and give some guidelines for designing such protocols.

Key words: Mobile Distributed Systems, Check pointing, Consistent Global State

1 Introduction

A mobile distributed computing system (MDCS) is a distributed system where some of the processes are running on mobile hosts (MHs). A mobile host communicates with other nodes of the distributed system via a special node called mobile support station (MSS). A cell is a geographical area around an MSS in which it can support an MH. An MH can change its geographical position freely from one cell to another or even to an area covered by no cell. An MSS has both wired and wireless links and acts as an interface between the static network and a part of the mobile network. Static network connects all MSSs. A static node that has no support to MH can be considered as an MSS with no MH [1]. Checkpointing is a technique that can be used for fault tolerance provisioning in distributed systems. A checkpoint is the state of a process on stable storage. In distributed system, a system state is said to be consistent if it contains no orphan message; i.e., a message whose receive event is recorded in the state, but its send event is lost. To recover from a failure, the system restarts its execution from a previous consistent global state saved on the stable storage. This saves all the computation done up to the last checkpointed state and only the computation done after that needs to be redone. In coordinated or synchronous checkpointing, processes take checkpoints in such a manner that the resulting global state is consistent. Mostly the coordinated checkpointing protocols follow the two-phase commit structure [4].

MDCSs raise many new issues such as mobility of nodes, low bandwidth of wireless channels, lack of stable storage on mobile nodes, disconnections, limited battery power and high failure rate of mobile nodes. These issues make traditional checkpointing algorithms unsuitable for checkpointing MDCSs [1], [3], [6]. A good checkpoint algorithm for MDCSs needs to have following characteristics [9]. The algorithm should be non-intrusive and should force minimum number of processes to take their local checkpoints. It should impose low memory overheads on MHs and low overheads on wireless channels. The disconnection of MHs should not lead to infinite wait state. The checkpointing algorithm should avoid awakening of an MH in doze mode operation.

Recently, minimum process coordinated checkpointing has been favoured for MDCSs [2], [3], [6], [7], [8], [12]. There are some non-blocking algorithms in which minimum processes are required to checkpoint; but some useless checkpoints are also taken which are discarded on commit [3], [6], [7]. In these algorithms, an effort is made to minimize the number of useless checkpoints. There are also some blocking minimum-process algorithms [2], [4], where no extra checkpoints are taken; efforts are made to minimize the blocking of processes. To maintain minimum process or non-intrusive property in coordinated checkpointing, we may require piggybacking of information onto normal messages [3], [6], [7], [8]. In all these protocols, an integer csn (checkpoint sequence number) is piggybacked onto normal messages. L. Kumar et al. [5] proposed an all process non-intrusive checkpointing protocol for distributed systems, where just one bit is piggybacked onto normal messages. This is done at the cost of vector transfers during checkpointing. The survey of checkpointing protocols for mobile distributed systems is given in [11], [14].

Cao and Singhal [3] achieved non-intrusiveness in minimum process algorithm by introducing the concept of mutable checkpoints. In their algorithm, a checkpointing tree is formed. During checkpointing, if a process, say P_i receives m from P_j such that P_j has taken some checkpoint in the current initiation before sending m, P_i may be forced to take a mutable checkpoint. Some of the mutable checkpoints are discarded on commit. Though only minimum number of processes takes permanent checkpoints but the actual number of processes that take checkpoints may be exceedingly high in some situations. L. Kumar et. al [6] and P. Kumar et. al [7] reduced the height of the checkpointing tree and the number of useless checkpoints by keeping non-intrusiveness intact, at the extra cost of collecting dependency vectors and computing the minimum set in the beginning. Minimum set is the set of interacting processes which need to take checkpoints along with the initiator. In other words, a process is in the minimum set only if the initiator process transitively depends upon it.

Koo and Toeg [4] proposed a minimum process and blocking algorithm, where each process uses monotonically increasing labels in its outgoing messages. Cao and Singhal [2] proposed minimum process blocking algorithm for MDCSs. They reduced blocking time as compared to [4] by collecting dependency vectors and finding the minimum set in the beginning. In their protocol, no information is piggybacked onto normal messages.

In coordinated checkpointing, if a single node fails to checkpoint in an initiation, the whole checkpointing effort goes waste. It becomes difficult for multiple MHs to checkpoint synchronously due to disconnections and unreliable wireless channels. MHs are prone to frequent failures, which will require frequent rollback of all processes. In the literature, there are hybrid checkpointing protocols, where fixed hosts checkpoint synchronously and MHs checkpoint independently [13]. These schemes give MHs autonomy in taking checkpoints. An MH can recover independently by using its recent checkpoint and message log without forcing other nodes to rollback.

2. Pitfalls in Existing Protocols

Prakash R. et al [9] proposed first nonblocking minimum process checkpointing protocol for MDCSs. Cao and Singhal [3] removed the inconsistencies in [9].

2.1 Problems in Minimum-Process Blocking Algorithm

(i) Problem 1

The algorithm may lead to inconsistency in some situations. Every process, say P_i, maintains a direct dependency bit vector (ddv_i[ ]) [j]=1) of length *n* for *n* processes. ddv_i[j]=1 implies P_i is directly dependent upon P_j. In Figure 1, at time t_1, P_2 initiates checkpointing process and sends the request for direct dependency vectors to P_1 and P_3. At time t_2, P_2 receives the dependency vectors from P_1 and P_3, computes minimum set [{P_2,P_3}in case of Figure 1], takes its own tentative checkpoint, and sends the checkpoint request along with minimum set to P_1 and P_3. A process is in minimum set if initiator process is transitively dependent upon it. It also needs to take a checkpoint along with the initiator. On receiving checkpoint request, P_1 does not take the tentative checkpoint, because it is not in the minimum set; P_3 is in the minimum set, therefore, it takes the tentative checkpoint. P_1 sends a message m_2 to P_3, before getting the request for direct dependency vector. P_2 receives m_2 during t_1 and t_2. This is the blocking time of P_2. In section 6 [2], the text says "More specifically, the MSSs cannot send messages during these 2*T_{static}. However, they can do other computations and even receive messages". Hence, P_3 can receive m_2 during this

blocking period. At time t_3, P_2 receives the responses, sends the commit request to P_3.
P_2 and P_3 discard their earlier permanent checkpoints i.e. $C_{2,0}$ and $C_{3,0}$, and convert their tentative checkpoints i.e. $C_{2,1}$ and $C_{3,1}$ into permanent. In this way, the recorded global state i.e. {$C_{1,0}$,$C_{2,1}$,$C_{3,1}$}is inconsistent due to m_2.

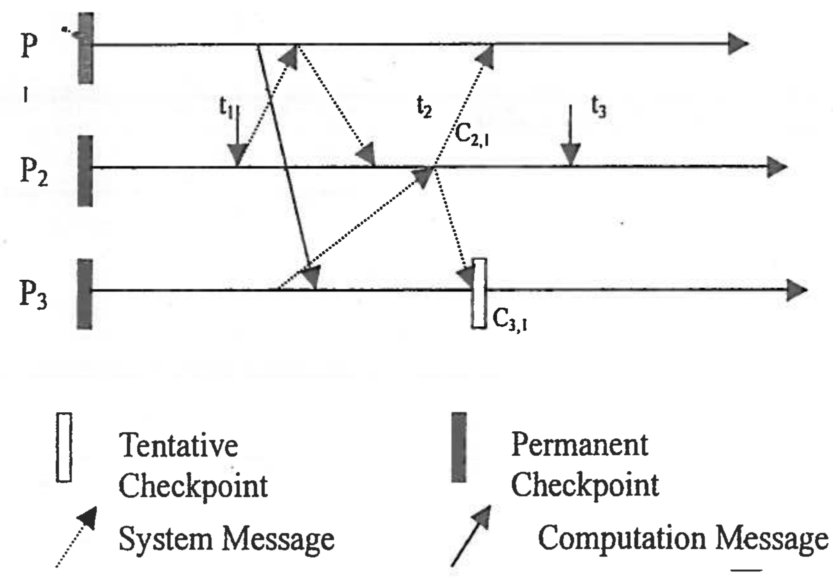

Figure 1

(ii) Problem 2

In Cao-Singhal algorithm [2], the number of processes, that take checkpoints, can be more than the minimum required as shown in Figure 2. P_1 sends m_2 before taking C_{11}, and P_2 receives m_2 after taking $C_{2,1}$. $\{C_{1,1}, C_{2,1}, C_{3,1}\}$ is a recovery line. According to their section 4.2, the dependency vectors, in the present case, are maintained as follows. When P_2 receives m_2, it sets $ddv_2[1]=1$. P_2 is not having sufficient information to conclude that P_1 has taken permanent checkpoint after sending m_2. $ddv_i[]$ is a direct dependency vector of P_i. Similarly, when P_2 receives m_1, it sets $ddv_2[3]=1$. At time t_1, P_2 initiates checkpointing and sends the request for direct dependency vectors to P_1 and P_3. At time t_2, P_2 computes minimum set [which in case of figure 3 is $\{P_1,P_2,P_3\}$], takes its own checkpoint, and sends checkpoint request to P_2 and P_3. P_2 and P_3 also take the checkpoints. In this case, P_2 is not dependent upon P_1 and P_1 is asked to take the checkpoint. Hence, the number of processes, that take the checkpoints, can be more than the minimum required.

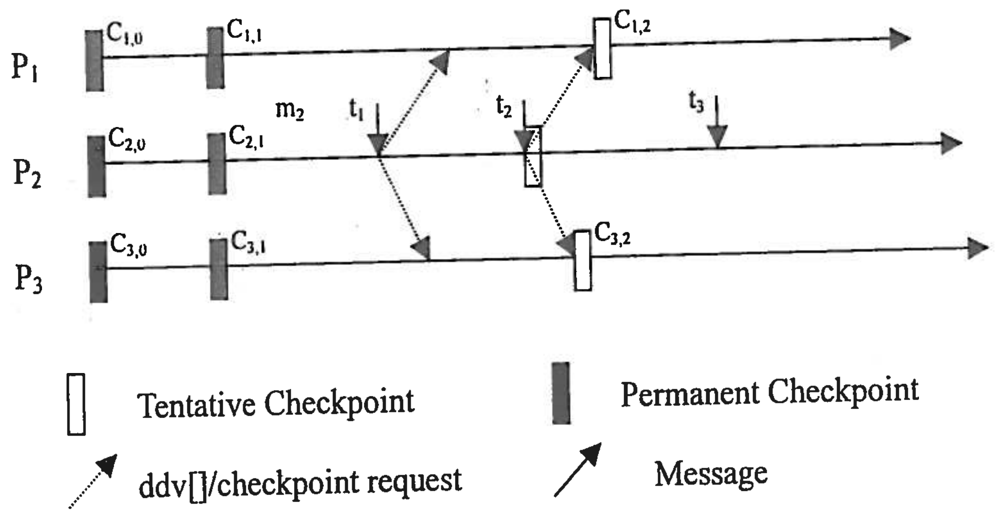

Figure 2

2.2 Problem in Singh-Cabillic [10] Algorithm

Singh and Cabillic [10] proposed a checkpointing algorithm for mobile computing environments on the basis of anti-message logging. This algorithm may lead to inconsistencies as follows.
In Figure 3, at time t_1, P_2 initiates checkpointing. Since, it has received m_1 and m_2 from P_1 and P_3, respectively, since last permanent checkpoint C_{21}; therefore, P_2 sends checkpoint request to P_1 and P_3. When P_1 receives the checkpoint request from P_2, it finds that it has not sent any message to P_2 since its last permanent checkpoint C_{12}. Therefore, P_1 discards the checkpoint request. P_3 receives m_3 without logging its anti-message. It should be noted that P_3 stores the anti-message of m_3 only if 1) P_3 has taken tentative checkpoint before receiving m_3 or 2) P_1 has taken tentative checkpoint before sending m_3. When P_3 receives the checkpoint request from P_2, it takes its tentative checkpoint C_{33}, because, it has sent m_2 to P_2 since its last permanent checkpoint C_{32}. After taking its tentative checkpoint, P_3 finds that it

has received m_3 from P_1 and P_1 has already been sent the checkpoint request; therefore, P_3 does not send the checkpoint request to P_1. In this way, $\{C_{12}, C_{22}, C_{33}\}$constitute a recovery line, where m_3 is an orphan message without its anti-message being logged at P_3. Hence, the algorithm [10] may lead to inconsistencies.

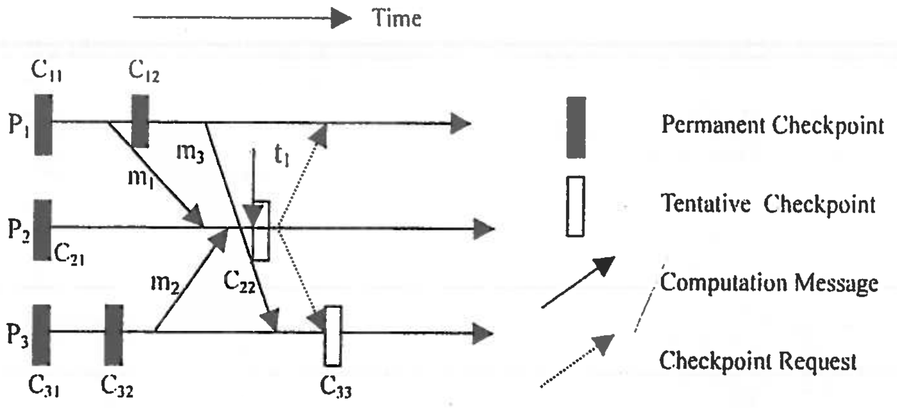

Figure 3

2.3 Problem in Cao-Singhal Algorithm

Cao and Singhal [3] achieved non-intrusiveness in minimum-process algorithm by introducing the concept of mutable checkpoints. In this algorithm, checkpoint initiator process (say P_i) sends the checkpoint request to P_j only if P_i receives m from P_j in the current checkpointing interval (CI). If P_j does not inherit the request, it simply ignores it. Otherwise, P_j takes its tentative checkpoint and propagates the request to P_k only if P_j receives m from P_k in the current CI. In this case, if P_j knows that some other process has already sent the checkpoint request to P_k and P_k is not going to inherit the current checkpoint request, then P_j does not send the checkpoint request to P_k. This process is continued till the checkpoint request reaches all the processes on which the initiator process transitively depends. Suppose, during checkpointing process, P_1 receives m from P_2. P_1 takes its mutable checkpoint before processing m only if the following conditions are met: (i) P_2 has taken some checkpoint in the current initiation before sending m (ii) P_1 has not taken any checkpoint in the current initiation (iii) P_1 has sent at least one message since its last permanent checkpoint. If P_1 takes mutable checkpoint and is not a member of the minimum set, it discards its checkpoint on commit.

We find the following observations in [3]:

(i) In this algorithm, multiple checkpoint requests may be sent between two MSSs as follows. Let us consider mobile distributed systems with two MSSs, say MSS_1 and MSS_2; where P_1 and P_2 are in the cell of MSS_1 and P_3 and P_4 are in the cell of MSS_2. Suppose, P_1 initiates checkpointing; and P_2 and P_3 are in its dependency set; i.e., P_1 is directly dependent upon P_2 and P_3. Similarly, P_4 is in the dependency set of P_2. In the existing protocol, P_1 sends checkpoint request to P_2 and P_3. After this, P_2 sends checkpoint request to P_4. In this way two messages are sent from MSS_1 to MSS_2. Although, there should be sent only one message. There is sufficient information at MSS_1 that P_1 is transitively dependent upon P_3 and P_4.

(ii) When P_i sends the checkpoint request to P_j, following scenarios are possible: (a) P_i knows that some other process has already

sent the checkpoint request to P_j (b) P_j is not in the minimum set (c) P_j discards the checkpoint request and P_j actually belongs to the minimum set.
(iii) When P_i sends a checkpoint request to P_j, it also piggybacks $csn_i[j]$ and a huge data structure MR[].
(iv) R_i[] maintains direct dependencies of P_i. In this algorithm, it is possible that $R_i[j]$ equals 1 and P_i is not directly dependent upon P_j for the current CI. For exactness, it is required that $R_i[j]=1$ only if P_i is directly dependent upon P_j. Hence, exact dependencies among processes are not maintained.
The useless checkpoint requests in above point [ii] are sent due to non-exact dependencies in point [iv]. The useless checkpoint requests are taken care of by sending sufficient information along with checkpoint requests in point [iii].
It is also interesting to note that the algorithm [3] does not lead to inconsistencies and the processes do not take unnecessary checkpoints due to the useless checkpoint requests. But the useless checkpoint requests and extra piggybacked information onto checkpoint requests increase the message complexity of the algorithm.

3 Discussions

In section 2.1, the problems, mentioned in point (i) and (ii), occur in the algorithm [2] due to following reasons. Exact dependencies among processes are not maintained. No information is piggybacked onto normal messages, which enables receiver process to decide whether it actually becomes dependent on the sender process after processing the message. For exactness, it is required: $R_i[j]=1$ only if P_i is directly dependent upon P_j. Most of the existing algorithms are unable to maintain exact dependencies among processes [2], [3]. Therefore, they lead to take some unnecessary checkpoints [2], send unnecessary checkpoint requests [3], piggyback complex data structure onto normal messages [3]. The algorithm proposed in [10], not only fails to maintain exact dependencies but also fails to take corrective measures to deal with non-exact dependencies. It should be noted the algorithm [3] takes corrective and costly measures to deal with incorrect dependencies. Therefore, we propose the following directions for designing minimum process checkpointing protocols for MDCS: (i) maintain correct dependencies among processes; (ii) piggyback necessary information onto normal messages to ignore the non-existent dependencies.

4. Conclusion

We have pointed out some complexities and inconsistencies in some minimum-process coordinated checkpointing protocol for mobile distributed systems. We investigate the cause of these problems and also propose the directions to avoid them. This work will enable researchers to design efficient checkpointing protocols for mobile distributed systems which are based on dependencies among processes.

References

[1] Acharya A. and Badrinath B. R., "Checkpointing Distributed Applications on Mobile Computers," Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems, pp. 73-80, September 1994.

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

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

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

[5] Lalit Kumar, M. Misra, R.C. Joshi, "Checkpointing in Distributed Computing Systems" Book Chapter "Concurrency in Dependable Computing", pp. 273-92, 2002.

[6] Lalit Kumar, M. Misra, R.C. Joshi, "Low overhead optimal checkpointing for mobile distributed systems" Proceedings. 19th International Conference on IEEE Data Engineering, pp 686 88, 2003.

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

[8] R K Chauhan, Parveen Kumar, Lalit Kumar, "A coordinated checkpointing protocol for mobile computing systems", International Journal of information and computing science, Vol 9 No. 1, 2006.

[9] Prakash R. and Singhal M., "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems," IEEE Transaction On Parallel and Distributed Systems, vol. 7, no. 10, pp. 1035-1048, October1996.

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

[11] Parveen Kumar, Lalit Kumar, R K Chauhan, "Synchronous Checkpointing Protocols for Mobile Distributed Systems: A Comparative Study", *International Journal of information and computing science*, Volume 8, No.2, 2005, pp 14-21.

[12] Parveen Kumar, Lalit Kumar, R K Chauhan, "A Hybrid Coordinated Checkpointing Protocol for Mobile Computing Systems", *IETE journal of research,* Vol 52, No. 2&3, pp 247-254, 2006.

[13] Lalit Kumar, Parveen Kumar, R K Chauhan, "Logging based Coordinated Checkpointing in Mobile Distributed Computing Systems", *IETE Journal of Research*, vol. 51, no. 6, pp. 485-490, 2005.

[14] Lalit Kumar, Parveen Kumar, R K Chauhan," Synchronous Checkpointing Algorithms for Mobile Distributed Systems", Journal of Multidisciplinary Engineering Technologies, Vol.1, No.2, pp 44-51,2006.

**Deptt. of Computer Sc & Applications*
Kurukshetra University, Kurukshetra-136119 (Haryana)

***Deptt. of Computer Sc & Engineering*
National Institute of Technology, Hamirpur (HP), 177005

pk223475@yahoo.com

BVCOE NEWS

IEEE STUDENT BRANCH INAUGURATION

The student branch of the IEEE was inaugurated at BVCOE New Delhi on 31st January, 2006 by Mr. Ajay Jain, Sr. Manager MOTOROLA. The inauguration ceremony was followed by a panel discussion. Mr. P V Ekande, Chairman IEEE , Delhi section, Prof. Balasubramanium, IIT Delhi, Prof. Bhatacharya, DCE, and Prof. Mini Thomas, Jamia Milia Engineering College.

ALUMINI MEET

Alumni meet of BVCOE, New Delhi was organized on 19th Feb, 2006. It concluded on a promise that the enlightened will be going on to the next era, with a positive attitude. With the assembly of the alumni, a good number of the present students and faculty members attended the cultural events and other activities.

HERTZ' 06

HERTZ' 06, a technical festival was organized by the Electrical and Electronics Engg. Deptt. on 7th March, 2006 at BVCOE New Delhi. Various events like technical paper presentation, LAN gaming, multimedia quiz, hardware design, project display, technical debate, robotics and GD etc took place. Mr B.S.Galgot, Vice President Havels India limited inaugurated the fest. Prof.G.S.Sandh, Director KJET was the guest of honour.

GENESIS 06

The Deptt. of Instrumentation and Control Engineering organized Genesis 06, a technical festival, on 10th March 2006 at BVCOE New Delhi. Invited talks were delivered by Mr.M.C. Manocha, Vice President, Reliance Energy and Mr.Arora from ISA. Expert lectures on Bio Medical Instrumentation, Project Exhibition , Paper Presentations. Quiz etc. were held during the technical festival.

National Conference on Information Technology and its application

The Department of Information Techmology organized a National Conference on Information Technology and its applications on 13th and 14th April 2006. The conference was inaugurated by Hon'ble chief minister of NCT of Delhi, Smt. Shiela Dixit.Dr. K K Aggarwal, Vice Chancellor, Guru Gobind Singh Indraprastha University (Delhi), was the guest of honour. The Deptt.of Science and Technology (Govt. of India) and AICTE sponsored the conference.

BV School of Multidisciplinary Research & Development (Recognized By Bharati Vidyapeeth Deemed University, Pune)

Bharati Vidyapeeth School of Multidisciplinary Research & Development is running EU-ASIA Link Project for the last 18 months. The project has four partners: Hunan University, China; Aalborg University, Denmark; Brunel University, U.K.; Bharati Vidyapeeth Deemed University, India. According to project implementation arrangement of the European Community funded Asia-link Project entitled "A Multidisciplinary Approach to Curriculum Development in Sustainable Built

Environment (Contract No. CN/ASIA-LINK/012 (93-520), the 3rd Workshop of the project was recently held at Brunel University, London, UK, during June 26 - 30, 2006. Dr. N. D. Kaushika and Mr. S. S. Mulik attended the workshop and presented the draft of following Educational Packages.

A. Solar Passive Building Technologies

B. Building and Sustainable Energy

C. Sustainable Building Materials

The contents of Sustainable Building Materials were modified & the contents of other educational packages were approved as such. And now the finished draft is to be completed by September 30, 2006. The meeting of the 4th Workshop will be held in India during March 5-9, 2007. Bharati Vidyapeeth College of Engineering, New Delhi on behalf of BV Deemed University, Pune will host the workshop. The meeting also discussed the mode of utilization of the sanctioned amount.

Following the above meeting Prof. N. D. Kaushika attended the Sixth IASTED-2006 International Conference on Communication Systems and Applications held at Banff, Alberta, Canada during July 3-5, 2006. He presented a three-hour tutorial entitled "VALUE ADDED SERVICES IN COMMUNICATION NETWORK". The summary content of the tutorial was published in the book abstracts as follows.

This tutorial will introduce the participants to new directions in value added services resulting from application of speech technology in 4 G communication networks. At the outset it reviews 4 G as a completed new fully IP based integrated system and network of networks achieved after convergence of wired and wireless networks as well as computers, consumer electronics and communication technology and several other convergences. The tutorial concludes with a discussion on assorted topics such as profiting from value added wireless services, SMS and ring tones through brain computer interface, Computer Telephony Interface (CTI), Interactive Voice Response (IVR), Automatic Speech based data query systems and multi-lingual conferencing over multimedia networks spread around the globe.

The group photo of EU-Asia Link Project Meeting at Brunel University, Uxbridge, UK.