

Fault Management in Mobile Computing

Business people on the go need portability and mobility in their computing environments. However, such mobile environments often suffer from transient faults. This article discusses how to manage faults in order to get better availability for small wireless devices.

By Goutam Kumar Saha

Mobile computing technology allows transmission of data via a computer system without being connected to a fixed physical link. Mobile data communication is a very important and rapidly evolving technology because it allows us to transmit data from remote locations to other fixed or remote locations. It solves the problem of business people on the move. Mobile computing is concerned with the connection of mobile or portable computers, and with the migration of software from a home platform to a remote one. We need access to computing and communications not only from our home base, but also while we are in transit and when we reach our destination.

The objective of such *transparent virtual networking* is to permit users and programs to be as effective as possible in environments of uncertain connectivity without changes to the manner in which they operate. Using diverse phones from everywhere in the world is not activity that could be called mobile computing because there is no computing or data processing involved.

Mobile computing consists of traveling people using portable wireless computing and communication devices connected to computer network infrastructure (e.g. Internet & wireless technologies) supporting global connectivity and remote computing. Portability, mobility, wireless communications, and system issues are the important technical challenges of mobile computing. Mobile computing faces the limitations of narrow bandwidth, limited computing power, small memory and battery capacity.

Fault Management

Fault tolerance is the ability of a system to perform its function correctly even in the presence of internal faults. The purpose of fault tolerance is to increase the dependability of a system. A complimentary but separate approach to increasing dependability is fault prevention. This consists of techniques, such as inspection, whose intent is to eliminate the circumstances by which faults arise.

Faults can be classified as *transient* or *permanent*. A transient fault will eventually disappear without any apparent intervention, whereas a permanent one will remain unless it is removed by some external agency. While it may seem that permanent faults are more severe, from an engineering perspective they are much easier to diagnose and handle. The intermittent transient faults that recur often unpredictably are the most problematic. Nowadays, mobile computing is a new software paradigm that is of prime interest in the Information Technology research community. This article looks at various fault tolerance issues applicable to mobile devices such as Personal Digital Assistants (PDAs).

The mobile computing environment is characterized by limited resource availability, high mobility, low bandwidth and frequent disconnection. Because of high mobility, network connectivity is often lost or degraded. Users may leave the area of network connectivity, or may enter into an area of higher interference. Mobile devices need to be lightweight and small in order to facilitate high mobility. This leads to the use of small batteries resulting in low available power. Processing of mobile computing applications is also halted due to such battery drain. Again, frequent disconnections can halt the applications. Since PDAs have small memories, they are not able to store much data to work upon during the time of disconnection. All such features make the mobile computing environment more susceptible to transient faults. Software faults are the results of problems in software running on PDAs, e.g., mobile crash. In this article, permanent software errors or bugs in program code, have not been considered, because they can not be corrected by external fault tolerant mechanisms. Only transient software faults, which exhibit the *fail – stop* model -- i.e., they do not occur at the same place again after a complete restart -- are considered. Such faults appear and disappear from time to time because of their transient nature.

Mobile systems are based on the concept of transience. The transient nature is also present in traditional distributed systems but only in the form of data. Whereas the mobile systems have extended this further with the advancement of wireless technology, mobile computing is entering the human world with a promise of providing computation ability while on move. Today handheld devices can perform various common applications of mobile computing such as communication with other devices, gathering of information, making financial transactions, and running downloaded audio and video applications.

Faults are an integral part of the world of wireless technology due to the aforementioned problems of frequent disconnections, narrow bandwidth, lack of resources and high mobility. Hence, the necessity of providing better fault-tolerant mechanisms for ensuring reliable working of mobile devices such as PDAs, can not be ignored. PDA is a lightweight, compact, and handheld device. Its applications include Web-browsing, e-mail, to-do list, watching downloaded videos. A device like infrared, bluetooth, or GSM is also associated for providing wireless network connection. PDAs have very limited resources. The normal battery life is from 8 hours to 1 week depending on work being done. At present processors used in PDAs are in the range of 400 MHz. PDAs have three kind of memories: RAM, ROM, and flash-based memory cards. Mobile computing requires an integrated solutions towards fault-tolerant mechanisms, for example battery power can not be saved if some check-pointing algorithm uses many system messages to

complete. It is required to provide a complete fault-tolerance solution for various applications of different natures.

Let us consider an example where an *mpeg* application is showing a video clip directly from a server and does not need check-pointing mechanisms. Instead it needs the mechanisms to deal with bandwidth variation (whereas a to-do list application requires check-pointing mechanisms). Various environment faults have also been considered. In the case of wireless networks, interaction between surrounding environment and wireless signals is very high. Such interaction results in interference and noise that eventually results in faults like reduction in bandwidth or packet losses. Such faults come under the category of environment faults. In this article, work has been carried out in order to provide availability for mobile computing applications running on PDAs. Availability is defined here in terms of providing fault-tolerance to running applications and enhancing resources for future mobile computing applications.

The following steps show how to manage various types of faults in a mobile computing environment in order to get better availability for mobile computing applications on mobile devices like PDAs. Various error recovery routines e.g., *Error_Routine_FEC*, *Error_Routine_Chp* etc., can be designed and invoked depending on the nature of various faults detected in such mobile computing environment for attaining fault tolerance.

If fault == *Packet Loss*, Then:

 Display "Nature of Fault is *Environment*"

 Display "Presence of Fault is *Transient*"

 Display "Suggested Solutions is *Forward Error Correction*"

 Run *Error_Routine_FEC*

Else If fault == *Mobile Crash*, Then:

 Display "Nature of Fault is *Software*"

 Display "Presence of Fault is *Transient*"

 Display "Suggested Solutions is *Checkpointing*".

 Run *Error_Routine_Chp*

Else If fault == *Transaction Failure*, Then:

 Display "Nature of Fault is *Software*"

 Display "Presence of Fault is *Transient*"

 Display "Suggested Solutions is *Checkpointing*".

 Run *Error_Routine_Chp*

Else If fault == *Out of Reach*, Then:

 Display "Nature of Fault is *Environment*"

 Display "Presence of Fault is *Transient*"

 Display "Suggested Solutions is *Hoarding*".

 Run *Error_Routine_Hrd*

Else If fault == *Bandwidth Reduction*, Then:
 Display "Nature of Fault is *Environment*"
 Display "Presence of Fault is *Transient*"
 Display "Suggested Solutions is *Prefetching , prenegotiation*".
 Run *Error_Routine_PP*

Else If fault == *Memory Capacity*, Then:
 Display "Nature of Fault is *Architectural*"
 Display "Presence of Fault is *Permanent*"
 Display "Suggested Solutions is *Prefetching , Buffer sharing*".
 Run *Error_Routine_PBShr*

Else If fault == *Problem in Battery*, Then:
 Display "Nature of Fault is *Architectural*"
 Display "Presence of Fault is *Permenent*"
 Display "Suggested Solutions is *Strategies for Power Saving*".
 Run *Error_Routine_SPWRSV*

End If

Conclusion

The proposed technique deals with an integrated approach towards better fault management for higher fault-tolerance as a whole. This approach is flexible and integrated. New mechanisms can also be integrated with this technique.

Further Readings

- [1] C.Y. Lin, S.Y. Kuo, and Y. Huang, "A checkpointing tool for palm operating system", *Proc. 2001 International Conference on Dependable Systems and Networks*, July 2001.
- [2] J.R. Lorch and A.J. Smith, "Software strategies for portable computer energy management", *IEEE Personal Communications magazine*, 5(3), pp 60-73, June 1998.
- [3] G.H. Forman and J. Zahorjan, "The challenges of mobile computing", *IEEE Computer*, 1994.
- [4] G. Cao and M. Singh, "Mutable Checkpoints: A new Checkpointing approach for mobile computing systems", *IEEE Trans. on Parallel and Distributed Systems*, 12(2), Feb 2001.

- [5] Goutam K Saha, "Transient Fault – Tolerant Mobile Agent in Mobile Computing", Paper No. TC-0157-0903 in *IEEE Transactions on Computers*, IEEE Computer Society Press, USA, 2003.
- [6] R. Prakash and M. Singhal, "Low cost checkpointing and failure recovery in mobile computing systems", *IEEE Trans. on Parallel and Distributed Systems*, 7(10), pp. 1035 – 1048 , 1996.

Goutam Kumar Saha is with the Centre for Development of Advanced Computing, Kolkata, India. Previously, he worked in LRDE, Defense Organization, Bangalore, India, as a scientist. His field of interest is transient software fault tolerance.

(Ubiquity, Volume 4, Issue 32, Oct. 8 - 14, 2003)