On Fault Tolerance, Performance, and Reliability for Wireless and Sensor Networks

CHEN Xinyu

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science & Engineering

On Fault Tolerance, Performance, and Reliability for Wireless and Sensor Networks

submitted by

CHEN Xinyu

for the degree of Doctor of Philosophy

at the Chinese University of Hong Kong

Abstract

The emerging mobile wireless environment poses exciting challenges for distributed fault-tolerant (FT) computing. This thesis develops a message logging and recovery protocol on the top of Wireless CORBA to complement FT–CORBA specified for wired networks. It employs the storage available at access bridge (AB) as the stable storage for logging messages and saving checkpoints on behalf of mobile hosts (MHs). Our approach engages both the quasi-sender-based and the receiver-based message logging techniques and conducts seamless handoff in the presence of failures.

In the proposed FT architecture, AB plays an essential role in relaying, logging, and redispatching messages, in performing handoffs, and in saving checkpoints. Messages will be queued at AB in the presence of MH failures and handoffs, thus AB becomes the performance bottleneck. Combining queueing model with five message scheduling strategies: the basic queueing model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model, we study the message sojourn time at AB under steady state.

Then we extend the analysis of the program execution time without and with checkpointing in the presence of MH failures from wired to wireless networks. Due to the underlying message-passing communication mechanism, we employ the number of received computational messages instead of time to indicate the completion of program execution at an MH. Handoff is another distinct factor that should be taken into consideration in mobile wireless environments. Three checkpointing strategies, deterministic, random, and time-based checkpointing, are investigated. In our approach, failures may occur during checkpointing and recovery periods.

Furthermore, we extend the traditional reliability analysis. Wireless networks inherit the unique handoff characteristic which leads to different communication structures of various types with a number of components and links. Therefore, the traditional definition of two-terminal reliability is not applicable anymore. We propose a new term, end-to-end mobile reliability, to integrate those different communication structures into one metric, which includes not only failure parameters but also service parameters. Nevertheless, it is still a monotonically decreasing function of time. With the proposed end-to-end mobile reliability, we could identify the reliability importance of imperfect components in wireless networks.

Finally, to obtain a long network lifetime without sacrificing crucial aspects of quality of service (area coverage, sensing reliability, and network connectivity) in wireless sensor networks, we present sensibility-based sleeping configuration protocols (SSCPs) with two sensing models: Boolean sensing model (BSM) and collaborative sensing model (CSM). Based on arc-coverage and Voronoi diagram with BSM, two sleeping candidate conditions to preserve k-coverage are developed respectively, which can deal with different sensing radii. As the CSM exploits the cooperation between adjacent sensors, which benefits to provide FT, we propose neighboring-sensor field sensibility (NSFS) to select sleeping-eligible sensors. With our protocols, redundant sensors are optionally identified and scheduled to sleep in order to extend the system lifetime while maintaining adequate sensor redundancy to tolerate sensor failures, energy depletions, and location error. Simulation results show that there is a tradeoff among energy saving, area coverage, and FT, which varies between different sensor configuration protocols and sleeping candidate conditions.

摘要

本論文討論無線網路和感測器網路的容錯、性能和可靠性。

在無線 CORBA 規範的基礎上，我們採用消息日誌和檢查點的機制發展了容錯無線 CORBA 架構。此架構支援移動單元（MH）的可靠交接（Handoff），能在移動單元和無線橋接器（AB）發生故障時恢復應用程式使其繼續運行。我們提出的新架構可做爲 FT－CORBA 規範的補充。

在提出的容錯無線 CORBA 中無線橋接器扮演很重要的角色。由於移動單元在發生故障或進行交接時不能接收消息，這些消息將會在無線橋接器中阻塞，所以需要分析無線橋接器的性能。我們結合排隊論和消息調度理論，構建出 5 種消息調度模型，推導和模擬在穩定狀態下消息的平均逗留時間。

隨後我們把在單台機器上程式運行時間的分析擴展到分布式的無線網路中。在單台機器上，程式運行是否結束是以無故障運行的時間來衡量的。然而由於網路鏈路和移動單元交接的影響，此結束條件不能應用於無線網路。我們將移動單元需要接收的消息個數作爲程式運行結束的條件。據此我們分析程式在採用 3 種不同的檢查點協議時的運行時間：確定的、隨機的和基於時間的。

接著我們通過提出的點對點的移動可靠性，將網路可靠性的分析從傳統的有線網路擴展到無線網路。點對點的

移動可靠性考慮無線網路特有的交接特性，集成由此導致的不同的通信模式。我們可以據此評估無線網路中各個組件的可靠性的重要程度。

最後，在保證區域覆蓋、感測可靠性和網路連通性的前提下，通過安排不同的感測器睡眠省電機制來延長感測器網路的工作時間。我們描述了布爾和協作兩種感測模型，同時基於弧覆蓋和 Voronoi 圖提出 3 個成爲睡眠候選人的條件。這些條件可應用於異構的感測器（不同的感測器可以使用不同的感測半徑），還考慮感測器的位置誤差。我們建議 3 種構造可靠的感測器網路的途逕：增加覆蓋度或減小睡眠配置時的通信半徑；適應性地安排各個感測器的睡眠；利用感測器之間的協作。

Acknowledgment

First of all, I would like to express my sincere gratitude towards my supervisor, Prof. Michael R. Lyu. My Ph.D. studies could never have been completed without his exceptional guidance and consistent support. His breadth of knowledge and his enthusiasm for research amaze and inspire me. Special thanks go to my thesis committee members, Prof. Jason Yi-Bing Lin, Prof. John C.S. Liu, and Prof. Jerome Chih-Hung Yen, for their many comments and feedback on my thesis work and research directions.

Time spent at CUHK would not have been interesting and enjoyable without the friendship provided by my officemates and friends: Hui Chen, Dr. Yan Ding, Jifu He, Dr. Kaizhu Huang, Mingfei Jiang, Wei Meng, Liang Wan, Guangyu Wang, and Wen Wu. I extend my gratitude to Xia Cai, Pik-Wah Chan, Steven Hoi, Xiaoqi Li, Yi Liu, Shi Lu, Cheuk-Han Ngai, Dr. Haixuan Yang, and Yangfan Zhou for their help and discussion in many aspects of my research work. It has been a great pleasure working with all of you. I would especially like to thank Dr. Qirong Deng for all his help in mathematical formulations. I also express my appreciation to the faculty and staff in the Department of Computer Science and Engineering.

Finally, this work is dedicated to my family. My parents taught me the value of knowledge, the joy of love, and the importance of family. They have stood by me in everything I have done, providing constant support, encouragement, and love.

Contents

Abstract **ii**

Acknowledgement **vii**

1 Introduction **1**

1.1 Wireless Infrastructure Network 2

1.1.1 Architecture of Wireless CORBA 3

1.1.2 Message Logging and Recovery in Wireless CORBA . . . 5

1.1.3 Message Queueing and Scheduling at Access Bridge . . . 6

1.1.4 Program Execution Time at Mobile Host 7

1.1.5 Reliability Analysis for Various Communication Schemes 9

1.2 Wireless Ad Hoc Sensor Network 10

1.2.1 Sensibility-Based Sleeping Configuration 13

1.3 Contributions of this Thesis 14

1.4 Organization of this Thesis 16

2 Background and Literature Review **18**

2.1 Fault Tolerance in Wireless Infrastructure Networks 18

2.2 Job Queueing and Scheduling 21

2.3 Program Execution Time 24

2.4 Network Reliability 28

2.5 Sleeping Configuration in Wireless Ad Hoc Sensor Networks . . 30
2.5.1 Area Coverage 30
2.5.2 Network Connectivity 33
2.5.3 Area Coverage and Network Connectivity 34
2.5.4 Voronoi Diagram 35

3 Message Logging and Recovery in Wireless CORBA 36
3.1 Fault Tolerance Model 36
3.1.1 Data Structures 38
3.1.2 Message Logging and Checkpointing 39
3.1.3 Mobile Host Handoff 41
3.1.4 Mobile Host Disconnection 43
3.1.5 Mobile Host Crash 45
3.1.6 Access Bridge Crash 47
3.2 Simulations and Evaluation 48
3.3 Summary 51

4 Message Queueing and Scheduling at Access Bridge 52
4.1 Mobile Host's State Transition 53
4.2 Message Sojourn Time 54
4.2.1 Basic Dispatch Model 55
4.2.2 Static Processor-Sharing Dispatch Model 56
4.2.3 Dynamic Processor-Sharing Dispatch Model 58
4.2.4 Cyclic Polling Dispatch Model 60
4.2.5 Feedback Dispatch Model 61
4.3 Simulations and Discussions 62
4.4 Summary 67

5 Program Execution Time at Mobile Host 68

5.1 Assumptions and Notations 69

5.2 Preliminary Execution Times 72

5.2.1 Handoff Time with Wireless Link Failures 72

5.2.2 Checkpointing Time with Handoffs and Wireless Link Failures 74

5.2.3 Recovery Time with Handoffs and MH and Wireless Link Failures 75

5.3 Deterministic Checkpointing Strategy 75

5.4 Random Checkpointing Strategy 81

5.5 Time-based Checkpointing Strategy 85

5.6 Comparisons and Discussions 87

5.7 Summary 94

6 Reliability Analysis for Various Communication Schemes 96

6.1 Definitions and Assumptions 97

6.2 End-to-end MR and MTTF Analysis 100

6.2.1 The MS (MH-SH) Scheme 100

6.2.2 The SM (SH-MH) Scheme 104

6.2.3 The MM (MH-MH) Scheme 111

6.2.4 General End-to-end MTTF 116

6.3 Summary 117

7 Sensibility-Based Sleeping Configuration in Sensor Networks 119

7.1 Assumptions and Problem Formulation 120

7.1.1 Boolean Sensing Model 121

7.1.2 Collaborative Sensing Model 122

7.1.3 Relations between the BSM and the CSM 124

7.1.4 Voronoi Diagram 125
7.2 Sleeping Candidate Conditions 125
7.2.1 Sleeping Candidate Condition for the BSM with Arc-Coverage 126
7.2.2 Sleeping Candidate Condition for the BSM with Voronoi Diagram 133
7.2.3 Sleeping Candidate Condition for the CSM 139
7.2.4 Location Error 141
7.2.5 Network Connectivity 143
7.3 Sensibility-Based Sleeping Configuration Protocol (SSCP) . . . 144
7.3.1 Round-Based Sleeping Configuration Protocol 145
7.3.2 Adaptive Sleeping Configuration Protocol 150
7.4 Simulations and Performance Evaluation 151
7.4.1 Configuration Protocols for Comparison 152
7.4.2 Parameters Setting 152
7.4.3 Experimental Results and Discussions 153
7.5 Summary 165

8 Conclusions and Future Directions 167

A List of Acronyms 171

B List of Notations 174

Bibliography 180

List of Figures

1.1 Architecture of wireless CORBA. 3
1.2 A wireless sensor network. 11
1.3 Thesis overview. 15

3.1 Fault-tolerant wireless CORBA architecture. 37
3.2 Normal invocation sequence with message logging. 40
3.3 Handoff procedure. 42
3.4 Access Bridge ORB. 44
3.5 Program execution time with and without checkpointing. 49
3.6 Program execution time vs. handoff. 50

4.1 MH's state transition. 53
4.2 Basic dispatch model. 55
4.3 Static processor-sharing dispatch model. 56
4.4 Head-of-the-line priority queue for static processor-sharing dispatch model. 57
4.5 Dynamic processor-sharing dispatch model. 59
4.6 Cyclic polling dispatch model. 60
4.7 Feedback dispatch model. 61
4.8 Expected message sojourn time vs. number of MHs. 63
4.9 Expected message sojourn time vs. message arrival rate. 64

4.10 Expected message sojourn time vs. MH's failure rate. 65
4.11 Expected message sojourn time vs. expected message dispatch requirement. 66

5.1 MH's state transition with checkpointing. 70
5.2 Average effectiveness vs. message number. 88
5.3 Average effectiveness vs. message arrival rate. 90
5.4 Average effectiveness vs. failure rate. 91
5.5 Average effectiveness vs. checkpoint taking time. 92
5.6 Average effectiveness vs. handoff rate. 93
5.7 Optimal checkpointing frequency. 93

6.1 System states in the MS scheme: (a) normal communication; (b) handoff procedure. 101
6.2 Markov model for the MS scheme. 101
6.3 State probabilities and reliability of the MS scheme. 102
6.4 End-to-end MTTF of the MS scheme: (a) failure parameters γ_m and γ_a; (b) service parameters ρ and η. 103
6.5 System states in the SM scheme: (c) location-querying; (d) normal communication; (e) handoff procedure; (f and g) location-forwarding. 104
6.6 Markov models for the SM scheme. 105
6.7 State probabilities and reliability of the SM scheme. 107
6.8 Reliability with location-forwarding strategies in the SM scheme. 108
6.9 End-to-end MTTF of the SM scheme vs. failure parameters: (a) γ_m and γ_a; (b) γ_s and γ_h. 109
6.10 End-to-end MTTF of the SM scheme vs. service parameters: (a) ρ and η; (b) ρ and ν. 109

6.11 RI of the SM scheme: (a) same failure rate and high handoff rate; (b) different failure rates and high handoff rate; (c) same failure rate and low handoff rate; (d) different failure rates and low handoff rate. 110

6.12 System states in the MM scheme: (h) location-querying; (i) normal communication; (j) MH_1 in handoff; (k) MH_2 in handoff; (l) both MH_1 and MH_2 in handoff; (m and q) location-forwarding; (n) location-querying and MH_2 in handoff; (o and r) location-forwarding and MH_1 in handoff; and (p) location-querying and MH_1 in handoff. 112

6.13 Markov models for the MM scheme (transitions to the failure state z is omitted). 113

6.14 State probabilities and reliability of the MM scheme. 115

6.15 General end-to-end MTTF vs. number of components. 116

7.1 Sponsored sensing region, arc and angle and covered sensing angle 127

7.2 Special cases of sponsored sensing region and arc 128

7.3 Derivation of the MPAC ξ_{ij} sponsored by sensor N_j to sensor N_i 129

7.4 Sponsored sensing arcs in a constrained deployment region . . . 132

7.5 Critical arcs in evaluating N_1's sleeping eligibility 133

7.6 Example of coverage boundary: N_1. 135

7.7 Example of sleeping-eligible sensor: N_1. 137

7.8 Example of necessary condition: N_1. 137

7.9 Scan region for sensor N_1. 140

7.10 The coverage relationship between a point and a sensor with location error. 142

7.11 Probability of coverage with location error. 143

7.12 Sensor status transition in SSCP. 146

7.13 An example of sleeping eligibility evaluation for the BSM with arc-coverage. 149
7.14 Percentage of sleeping sensors vs. communication radius cr. . . 154
7.15 Number of working vs. deployed sensors. 156
7.16 Energy consumption for configuration. 157
7.17 Average number of neighbors. 158
7.18 Field sensibility distribution. 159
7.19 Percentage of sleeping sensors vs. sensibility threshold ϵ_s. 159
7.20 Percentage of sleeping sensor with *SscpAcCa*. 160
7.21 Deviation of location. 161
7.22 χ-coverage accumulated time vs. MTTF when $\epsilon = 1$. 162
7.23 χ-coverage accumulated time vs. MTTF with FT approaches when $\epsilon = 1$. 164
7.24 Percentage of live sensors vs. time when the MTTF is 4000s and $\epsilon = 1$. 165

Chapter 1

Introduction

Advances in wireless networking technology and portable information appliances have brought about two implementations of wireless networks defined by *IEEE 802.11*: *infrastructure network* and *ad hoc network* [10]. In an infrastructure network, there are at least one Access Point (AP) connected to the wired network infrastructure and a set of wireless terminal devices. The AP acts as a message transceiver for the wireless network, aggregating access for multiple terminal devices onto the wired network. An ad hoc network is composed solely of wireless terminal devices within mutual communication range of each other without intermediary devices, such as AP. It is typically created in a spontaneous manner. If a wireless terminal device has the capability to measure a physical attribute or detect a physical event in the environment, a set of such devices forms a *wireless ad hoc sensor network.* New applications arise from wireless terminal entities interacting and collaborating towards a common goal. For example, wireless networks can be designed for supporting crew-computing tools aboard the International Space Station [7]; the planetary exploration may also employ mobile wireless networks as its communication system architecture [6]; in the battlefield, a general can gather real-time information from his soldiers and send commands to them; furthermore, the sensing capability built in wireless devices provides surveillance, reconnaissance, and tracking.

Experience reveals that components of wireless infrastructure and ad hoc networks are subjected to faults. Therefore, non-desired behaviors of networks can occur while being utilized, which interrupt or fail the delivery of specified services to network users. We know that *reliability* is a measure of the durability of a component or system delivering service over time. In order to provide reliability despite the presence of network component faults, mechanisms for *fault tolerance* (FT) must be adopted. FT is the ability of a system to continue providing its specified service despite component faults [77]. Tolerance to fault is generally based on *redundancy* techniques [48], which are the effective deployment and utilization of extra resources (time and/or space) to detect, correct, or mask effects of faults. Assessment of performance and reliability is a key step in the design, analysis, and tuning of FT computer systems [111]. Could additional overhead incurred by FT mechanics make the performance worse? Could we get a performance improvement just by changing the scheduling of jobs? Would an increase in performance outweigh the decrease in reliability?

Therefore, this thesis focuses on FT, performance, and reliability for both types of wireless networks: infrastructure network and ad hoc sensor network.

1.1 Wireless Infrastructure Network

In a wireless infrastructure network, users usually carry wireless terminal devices to move around, thus creating a new paradigm of decentralized computing, called mobile computing [60]. A mobile computing system is considered as an extension of traditional distributed systems, in which much of the action takes place at the middleware level. *Middleware* is a software layer that connects two otherwise separate applications and aims to resolve heterogeneity and distribution [41]. CORBA (Common Object Request Broker Architecture) [99], which is specified by the Object Management Group (OMG), is one of the most popular middlewares nowadays. CORBA provides portability, location transparency, and interoperability of applications across heterogeneous platforms (hardware architectures, operating systems, and programming languages) [92]. To support wireless access and terminal mobility in CORBA, the OMG has also published Wireless CORBA specification [100], which we employ in this thesis as a typical prototype for wireless infrastructure networks [110, 131] to build a fault-tolerant (FT) [1] wireless system, and then to conduct performance and reliability analysis and evaluation.

1.1.1 Architecture of Wireless CORBA

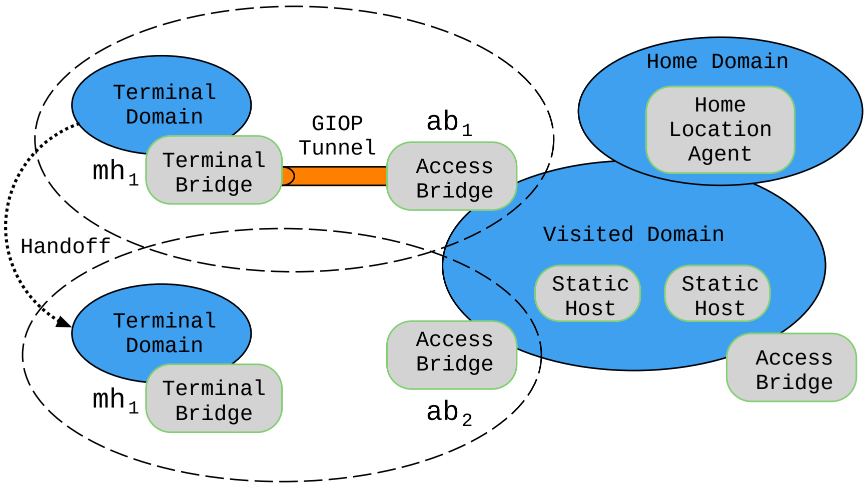


Figure 1.1: Architecture of wireless CORBA.

Figure 1.1 shows the wireless CORBA architecture, which identifies three different domains: Terminal Domain, Visited Domain, and Home Domain, and four main components excluding links: Terminal Bridge, Access Bridge, Home

[1]In this thesis, FT stands for both fault tolerance and fault-tolerant, which one is utilized depends on the context.

Location Agent, and Static Host.

- A *Terminal Domain* is a *Mobile Host* (MH) which accesses networks through a wireless network interface, and keeps network connections while roaming in wireless environments. The hosted *Terminal Bridge* (TB) on an MH is an object invocation proxy through which the CORBA objects on the MH can communicate with other objects.

- The *Visited Domain* is a wired network environment which contains many *Static Hosts* (SHs) and several *Access Bridges* (ABs). An SH is a common and stationary network node. An AB is located between MHs and SHs or other ABs to relay messages for its associated MHs. It is deployed in a wired network, but contains both wired and wireless network interfaces;

- The *Home Domain* is composed of the *Home Location Agent* (HLA) which keeps track of the locations of its registered MHs when they move around, and provides operations to query an MH's location.

An AB resides on an AP (or mobile support station) providing a constrained geographical communication cell, plotted as dashed circle in Figure 1.1, within which it can communicate with MHs directly. Multiple ABs provide a connected cells, allowing an MH to roam from one cell to another while maintaining network connections by a procedure called *handoff* (or *handover*) when the MH moves across the borders of the geographical cells. The handoff procedure occurs between an MH's new AB and its old AB.

All hosts communicate with each other by messages only. The GIOP (General Inter-ORB Protocol) tunnel is the communication channel between an AB and a TB, through which the GTP (GIOP Tunneling Protocol) messages are transmitted. The GTP messages can be classified into two categories: *control* message and *computational* message. No messages can be exchanged directly among MHs, even if they stay in the same cell. All messages to and from an MH are relayed by its currently associated AB. During handoff, no computational messages can be transmitted between MHs and ABs.

1.1.2 Message Logging and Recovery in Wireless CORBA

With wireless infrastructure networks, mobile computing enables users or explorers to access and exchange information while allowing them to roam around in mobile environments. This flexibility, however, causes more probable physical damage to MHs [94]. In addition, MHs contain low battery power, and wireless links suffer limited bandwidth and long transfer delay, which make transient failures more likely. Moreover, MHs may even disconnect from the hosting networks intermittently [108]. Finally, wireless systems are more often subjected to environmental conditions which can cause loss of communications or data [72]. All these call for a FT mobile computing system, especially to build a dependable battlefield wireless network as a system failure may cause loss of human lives.

Recently the OMG have specified FT–CORBA [99] as a standard to preserve the delivery of correct service in the presence of active faults. Based on entity redundancy, FT–CORBA employs three replication styles: cold passive, warm passive, and active replications. Logging and checkpointing mechanisms record messages and entity states in system logs. However, all these are intended for wired networks. This thesis proposes a message logging and recovery protocol on the top of wireless CORBA and FT–CORBA architectures [24]. An MH typically has a slow processor and small memory, and is commonly battery powered. Therefore, the MH itself cannot be considered suitable for the kind of stable storage that is usually performed by SHs. As all messages exchanged between an MH and other hosts should be forwarded by the MH's associated AB, the storage available at the AB is employed as the stable storage to log messages and *checkpoints* (saved states) on behalf of the MH [1, 94, 103, 105]. Both the quasi-sender-based and the receiver-based logging techniques are engaged in our approach. During failure-free execution, an MH takes checkpoints and sends the checkpoints to its currently connected AB. The AB also logs messages relayed by itself. After an MH failure, its associated AB has to collect those scattered checkpoints and message logs due to the movement of the MH, and resends these information to recover the MH's state. The AB hides MH disconnection from other network hosts [137] and makes a seamless handoff when FT properties are called upon. We also discuss how to tolerate MH disconnection, MH crash, and AB crash. After that, a simulation model is constructed to evaluate our proposed scheme.

1.1.3 Message Queueing and Scheduling at Access Bridge

From the above description, we learn that AB plays an essential role in providing FT for wireless CORBA. An AB is a message-relay station which disseminates messages between MHs and SHs. If an MH is in handoff, no computational messages can be forwarded to it, and messages received during handoff should be queued and buffered at the AB. The more probable failures also cause messages to be queued at the AB during the MH's recovery period. The AB, thus, becomes a bottleneck in improving the performance of FT in mobile wireless environments. Therefore, it is essential to study the performance of the AB, such as the expected message sojourn time, in the presence of failures and handoffs of MHs. As different message scheduling strategies should demonstrate various effects on the expected message sojourn time, in this thesis we consider five message dispatch strategies at AB: the basic queuing model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model [27]. We derive the expected message sojourn times under steady state for the static and the dynamic processor-sharing models and simulate all the five models. We perform comparisons among their expected values, discuss the similarities and differences among these models, and determine whether they are suitable to be engaged as a message distribution strategy at AB for wireless infrastructure networks.

1.1.4 Program Execution Time at Mobile Host

Analyses of the program execution time with and without checkpointing [40] in the presence of failures on a stand-alone host have been conducted by many researchers [39, 74, 96, 125]. In all previous work, it has been assumed that the required program execution time without failures is specified at first, after which the actual total program execution time is derived. Utilizing time to indicate the completion of program execution is feasible for SHs not engaged in distributed computing. However, mobile computing introduces information exchange between hosts; thus, the program execution time is not only controlled by the host itself, but also affected by other hosts and network conditions, such as the states of links. Due to the unpredictable nature of these conditions, the execution time is no longer suitable to be utilized as the given parameter to indicate the completion of a program. Moreover, mobile wireless environments introduce some other factors, such as random handoffs, which also interrupt the program execution at an MH.

With an information processing system built on the top of a mobile wireless environment, a mobile user can process debit/credit transactions, pay utility bills, make airline reservations, and carry out other transactions without being subjected to any geographical constraints [44]. The information processing system is essentially a distributed client/server system where a client at an MH interacts with a fixed server to complete a transaction. As the underlying communication mechanism in mobile computing is a message-passing system, a transaction conducted at an MH will proceed when it receives a specified message. Usually, a transaction has a dedicated number of phases that are demarcated by specified messages [129]; therefore, we may employ the number of specified messages to indicate the completion of transaction execution at an MH. Another reason to utilize the number of messages is that the computation at an MH may not be continuous since the transaction is forced to wait for the arrival of its predefined informative messages to proceed. The computation is actually determined by the arrival events of messages. We have stated in Subsection 1.1.1 that messages exchanged between MHs and ABs are GTP messages, which can be classified as control messages and computational messages. Control messages carry control commands and instructions, but they do not enable the computation to proceed. Computational messages, on the other hand, are the messages that the transaction at an MH should receive in order to accomplish its assigned task. If the transaction receives its last expected computational message, all phases in this transaction have been fulfilled, thus it is accomplished successfully. As a result, we assume here that if the transaction at an MH continuously receives n computational messages, it will terminate successfully. From this point on, we will refer to "computational messages" as simply "messages", and use "transaction" and "program" interchangeably.

Failures to an MH cause the current computational state to be corrupted. To recover the state, the MH should re-receive some or all messages that were received before the failure; this will prolong the total program execution time.

Without checkpointing, the program needs to recover its state from its initial start point; while with checkpointing, the program may restart from its most recent checkpoint and only needs to re-receive messages after that checkpoint. Therefore, the benefit of checkpointing comes from the reduction in the amount of rollback [46]. However, checkpointing itself introduces an overhead as no computational messages can be received during checkpointing, and the subsequent logged message replaying postpones the program completion. So there must exist a trade-off between checkpointing effectiveness and overhead, as well as an optimal checkpoint quantity to minimize the total execution time. In this thesis, we analyze the program execution time with various checkpointing strategies: deterministic checkpointing, random checkpointing, and time-based checkpointing. A checkpointing strategy is a rule that determines when to save the program's state on stable storage [34]. We derive the Laplace-Stieltjes Transform (LST) of the cumulative distribution function (c.d.f.) of the program execution time and its expectation for these three strategies [25, 28]. Different checkpointing strategies may demonstrate different performance behaviors under different conditions; therefore, variations of the program execution time with different parameters are demonstrated through a number of assessments.

1.1.5 Reliability Analysis for Various Communication Schemes

For a wireless CORBA network to be functional, its engaged components must be fit for service. Unfortunately, this is not always the case, because these components may suffer failures, and wired paths and wireless links may not be reliable as mentioned before. Therefore, we need a mechanism to assess the reliability of wireless infrastructure networks. Network-reliability analysis has long been an important area of research for wired networks [2, 64, 73, 78, 113] but not for wireless networks. Furthermore, the reliability issue for wireless CORBA is quite different from that for wired networks, as wireless CORBA introduce a unique feature called handoff. The handoff operation causes the existing communication structure in end-to-end communications to change with the MH's movement, i.e., at different time periods, different components are engaged in node-pair communications. Thus, the traditional two-terminal reliability defined in wired networks [118] is not suitable anymore. This thesis introduces a new concept, end-to-end mobile reliability, to define the reliability metric in wireless networks [26, 30], which not only keeps the monotonously decreasing characteristic of reliability but includes the mobility nature in the system. Different effects imposed by component failure parameters and mobile service parameters are given through numerical examples. To observe the gain in reliability improvement, the reliability importance (RI) of imperfect components are also evaluated.

1.2 Wireless Ad Hoc Sensor Network

The second implementation of wireless networks is wireless ad hoc sensor networks which are being increasingly deployed to perform certain tasks, such as sensing, tracking, measurement, and surveillance, as shown in Figure 1.2. The sensors, serving as the nodes in this kind of network, are tiny power-constrained devices, which connect together through short-range radio transmission and form an ad hoc network. By intelligently combining each sensor's reported data, an end-user can remotely monitors events in the deployment region. The monitoring and surveillance characteristics of a wireless sensor network require that every point in the region of interest should be sensed with given parameters by the cooperation of deployed sensors; otherwise, an

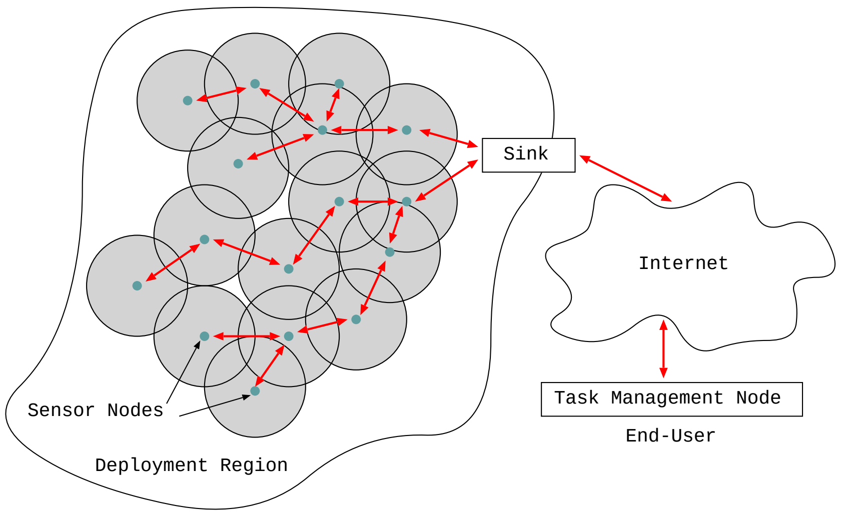


Figure 1.2: A wireless sensor network.

event occurring at the under-monitored points will not be detected. This is the coverage issue, one of the fundamental measures for quality of service of a wireless sensor network.

To preserve the coverage requirement, the network should sustain a long lifetime without sacrificing the system's reliability. However, as wireless sensors are microelectronic devices, the energy source provided for them is usually battery power, which has not yet reached the stage for sensors to operate for a long time without recharging or replacement. Furthermore, the unattended nature of sensors and hostile sensing environments make manual battery recharging or replacement undesirable or impossible [127]. As a result, finding ways to prolong the functional lifetime both of individual sensors and of the network is an important challenge. We know that if a sensor is frequently alternating between an active and an inactive state, its battery life will be extended [120]. From this observation, sleeping configuration has been proposed as a promising way to extend network lifetime by alternately activating only a subset of sensors and scheduling others to sleep according to some heuristic schemes while providing sufficient coverage in a geographic region.

Besides the coverage problem, sensors may fail or be blocked due to physical damage or environmental interference. The failure of sensors may produce some void areas that do not satisfy the coverage requirement. Therefore, another important design issue is to sustain sensor network functionality without any interruption due to sensor failure; this is termed the reliability or FT issue [4, 85]. The checkpointing and message logging mechanisms utilized for wireless infrastructure networks are not suitable for sensor networks. Moreover, the location error is introduced when the position of sensors cannot be engineered or predetermined with random deployment. One way to address these challenging problems is by deploying sensors densely. In a densely distributed sensor network, the system relies on the collective behavior of sensors to function reliably [139], i.e., it is the number, not the capability of each individual sensor, that really matters. But having too many sensors working at the same time increases the probability of packet collision, thus reducing the network throughput. Therefore, on the one hand, a sleeping configuration protocol should find as many sleeping-eligible sensors as possible to prolong network lifetime and to reduce packet collision; on the other hand, it should still retain enough redundancy to construct dependable sensor networks.

Finally, there is a scalability challenge associated with a high density of sensors when achieving the desired area coverage and robustness. Sensor networks are constructed with multi-hop communications, because generally using several short intermediate hops to send data is more energy efficient than using one longer hop [66, 91]. In addition, communication expends more energy than computation. This implies that each sensor itself must configure its own operational mode adaptively based on information about its neighborhood, not on the complete information about the deployment region.

1.2.1 Sensibility-Based Sleeping Configuration

In response to all the aforementioned requirements (maintaining area coverage, extending system lifetime, tolerating sensor failures and location error, and achieving scalability), we present a sensibility-based sleeping configuration protocol (SSCP) that is fully decentralized and localized with three sleeping candidate conditions. Two sensing models, Boolean sensing model (BSM) and collaborative sensing model (CSM), are investigated in our SSCP.

A number of sleeping configuration protocols have been proposed [58, 126, 133, 136, 143] in the literature. They have so far been based on the BSM, which assumes that each sensor has a certain sensing range, and a sensor can only detect the occurrences of events within its sensing range; it does not provide any sensibility out of this range. All these investigations are constructed on the basis of determining a sensor's sleeping eligibility with the coverage of sensing disks [126, 136], the coverage of sensing perimeters [58], or the coverage of intersection points of sensing perimeters [133, 143]. With the BSM, we propose a sleeping candidate condition called minimum partial arc-coverage (MPAC) based on the coverage of sensing perimeters. The proposed MPAC can deal with sensors with different sensing ranges, and can satisfy k-coverage requirement which indicates that every point in the deployment region is covered by at least k nodes. In addition, we develop a sleeping candidate condition with the property of Voronoi diagram [101].

Although the BSM allows a geometric treatment of the coverage problem, it misses the attenuation behavior of signals and ignores the collaboration between adjacent sensors in performing area sensing and monitoring. The CSM captures the fact that signals emitted by a target of interest decay over the distance of propagation, which is more realistic. With the CSM, we propose the neighboring-sensor field sensibility (NSFS) to provide dependable configurations for tolerating sensor failures, energy depletions, and location error by exploiting the cooperation between adjacent sensors.

In order to conserve energy, each sensor autonomously determines its own status (WORKING or SLEEPING) by utilizing partial sensor distribution information obtained through communications with its local neighbors. This property enables the SSCP to scale to large networks.

Simulations with ns-2 [42] show more sensors can be scheduled to sleep in the BSM than in the CSM, which is beneficial to prolonging system lifetime and reducing packet transmission collision. However, as too many sleeping sensors reduces the sensor redundancy which is crucial to tolerating sensor failures, energy depletions, and location error, the BSM will perform worse than the CSM with the accumulated coverage time. Therefore, a trade-off exists between energy conservation, area coverage, and FT. Three effective approaches to build dependable wireless sensor networks are suggested: increasing the required degree of coverage or reducing the communication radius during sleeping configuration, configuring sensor sleeping adaptively, and utilizing the cooperation between neighboring sensors.

1.3 Contributions of this Thesis

In this thesis, we conduct some investigations in FT, performance, and reliability issues for wireless infrastructure and ad hoc sensor networks, as shown in Figure 1.3:

- Build a FT architecture for wireless CORBA through message logging and recovery. Approaches for support reliable handoff, for hiding MH disconnection from other hosts, and for tolerating MH and AB crashes are

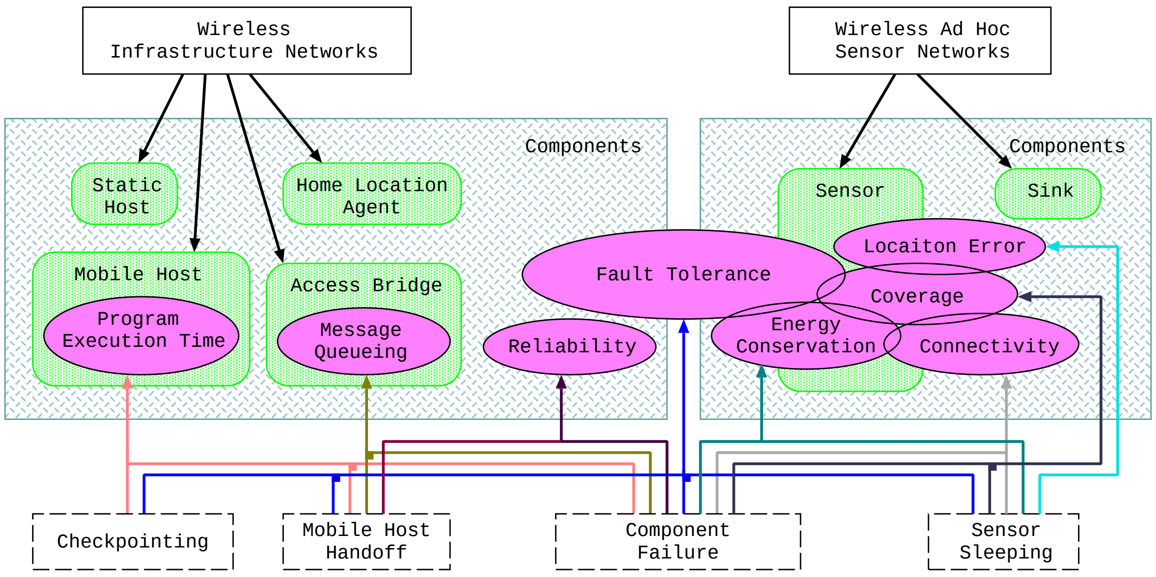


Figure 1.3: Thesis overview.

exploited. It supplements to build FT-CORBA in wireless environments.

- Innovatively combine queueing model with five message scheduling strategies at AB in the presence of MH failures and handoffs, and analyze and simulate their expected message sojourn times, as AB is a bottleneck of message dispatching in the proposed FT wireless CORBA.

- Extend the previous analysis work on the program execution time without and with checkpointing from a stand-alone host to an MH by engaging the number of received messages as the program execution termination condition; Furthermore, three checkpointing strategies are evaluated: deterministic checkpointing, random checkpointing, and time-based checkpointing.

- Integrate the mobility nature (handoff) of a wireless infrastructure network with the reliability evaluation for different communication schemes by our firstly proposed end-to-end mobile reliability, with which we could identify the reliability importance of imperfect components in wireless networks.

- Investigate two sensing models, BSM and CSM, in wireless ad hoc sensor networks, and propose three sleeping candidate conditions with arc coverage and Voronoi diagram to conserve sensor energy while preserving redundancy to tolerate sensor failures and location error. Two sleeping configuration protocols, round-based and adaptive sleeping, to configure distributed sensors are developed.

Overall, components are failure-prone in wireless infrastructure and sensor networks; therefore, we need to exploit FT in both networks. In addition, their corresponding performance and reliability issues should be investigated. However, the roles of these two networks are quite different: One is providing user mobility in wireless infrastructure networks and the other is monitoring area events in wireless sensor networks. As a result, the approaches to FT and the correlative measures for performance and reliability should be discussed in different ways to meet their corresponding missions.

1.4 Organization of this Thesis

The rest of this thesis is organized as follows: In the next chapter, a detailed review on FT, performance, and reliability issues in wired, wireless, and sensor networks are given. Chapter 3 builds a FT wireless CORBA with message logging and recovery. Chapter 4 analyzes the message dispatch strategies and their corresponding message sojourn times at AB in the presence of MH failures and handoffs. The program execution times at a failure-prone MH with three checkpointing strategies are developed in Chapter 5. Chapter 6 gives reliability analysis for various communication schemes. Chapter 7 presents the sensibility-based sleeping configuration for wireless sensor networks. Finally, Chapter 8 summarizes the thesis and discusses future directions.

□ **End of chapter.**

Chapter 2

Background and Literature Review

In this chapter, we conduct a detailed review on fault tolerance, performance, and reliability issues in wireless infrastructure and ad hoc sensor networks.

2.1 Fault Tolerance in Wireless Infrastructure Networks

Fault tolerance (FT) is the ability of a system to continue providing its specified service despite component failures. It is carried out via error detection and system recovery [12]. One technique of system recovery is *rollback recovery* which periodically saves system states during failure-free execution on stable storage and restarts a failed system component from one of its saved states [40]. Each of the saved states is called a *checkpoint*. In distributed systems, such as CORBA and wireless CORBA, messages complicate rollback recovery as messages induce inter-component dependencies. In addition to take a checkpoint, the messages transmitted may also be logged, thus introducing log-based rollback recovery. Checkpointing and message logging for wired networks have been studied extensively [9, 35, 40, 61, 87]. Most recently, some work has been conducted in providing FT for wireless environments.

Neves and Fuchs [43, 94] developed an adaptive and coordinated checkpointing protocol by saving consistent global states through local timers or the number of checkpoints piggybacked in computational messages. The time to next checkpoint is also piggybacked to approximately synchronize those local timers. The protocol logs in-transit messages when taking checkpoints at the sender. Hard checkpoints are saved on stable storage, and soft checkpoints are saved locally on MHs. Based on this checkpointing protocol, Ssu [122] proposed a leasing mechanism to manage storage for checkpoints and to dynamically adjust locations used to store checkpoints for reducing checkpoint transmission overhead. Instead, our protocol (Chapter 3) engages uncoordinated checkpointing to reduce the number and size of messages transmitted through wireless links. No messages will be logged on MHs to decrease power consumption and storage utilization.

Pradhan *et al.* [71, 105] described recovery schemes as a combination of state saving and handoff strategies. The state saving strategies include *no logging* and *logging*, and the handoff strategies include *pessimistic*, *lazy*, and *trickle* handoffs. It engages "local" AB (the MH's currently associated AB when checkpointing) as the stable storage, which implies that successive checkpoints of an MH may be scattered at different ABs due to MH movement. Our checkpointing and message logging protocol also choose the currently connected AB as the stable storage. In the pessimistic strategy, a checkpoint is transferred to the new AB during handoff. The lazy strategy creates a linked list of ABs visited by the MH, instead of transferring checkpoint and message log in each handoff. This strategy reduces the network overhead during handoff; however, upon an MH failure, its most recent checkpoint may be far away from the current AB, thus increasing the recovery overhead. The trickle strategy adds a distance constraint condition to ensure that the checkpoint and message log are always at most one-hop from the current AB. In our protocol, we adopt a handoff strategy like the trickle one, but we employ an adaptive, dedicated, and separate thread to collect the last checkpoint and the successive message log.

Park and Yeom [103] developed an asynchronous recovery scheme based on optimistic message logging, in which the AB performs logging and dependency tracking. The messages exchanged between ABs carry vector clocks for asynchronous recovery. But traced dependency information may be imprecise that leads to unnecessary rollback of MHs after a failure. We choose pessimistic message logging to avoid large number of control message interchange and rollback propagation. The authors also proposed a movement scheme for managing checkpoint and message log [102]. Only when an MH moves out of a predefined distance range or when the number of handoffs exceeds a threshold, its corresponding checkpoint and message log are collected to a nearby AB.

Juang [63] employed a matrix in each AB to maintain the dependency relationship among engaged MHs and SHs, by which an independent checkpointing protocol was developed. Upon a failure, a host needs to rollback only once and can immediately resume its operation without waiting for any coordination messages from other hosts. Cao and Singhal [16] introduced the concept of mutable checkpoint, which need not be saved on stable storage and can be saved anywhere, such as the main memory or local disk of MHs, so that avoiding the overhead of transferring large amounts of checkpoint information to the stable storage at ABs over the wireless network.

Acharya *et al.* [1] presented a checkpoint scheme in which each MH checkpoints its local state whenever it receives a message after sending a message, performs a handoff operation, or prior to disconnecting from its associated AB. Yao *et al.* [137] proposed a proxy-based recovery for applications on MHs. The proxy transparently monitors an MH's interactions with other hosts and maintains a copy of the MH's state. They also described a receiver-based pessimistic message logging protocol in [138]. Higaki and Takizawa [56] described a hybrid checkpointing protocol in which SHs take coordinated checkpoints, and MHs take uncoordinated checkpoints. When an MH moves between cells covered by ABs, it leaves an agent process on each AB to help its recovery upon failures. Some communication-induced checkpointing protocols have also been proposed, which allow checkpoints to be taken asynchronously [3, 80, 106].

To tolerate AB failures, Alagra *et al.* [5] utilized pessimistic and optimistic replication strategies by allowing MH move to a replicated AB, or by designing a network to cover each MH with more than one AB. Furthermore, Khan and Abd-El-Barr [65] utilized fuzzy logic to build a hybrid strategy by evaluating the best ratio of pessimistic to optimistic replicated ABs.

Ruggaber and Seitz [108, 109] introduced Π^2, a proxy platform for CORBA-based applications in the nomadic environments. It splits the connections between an MH and an SH by a proxy to avoid suffering from sudden disconnections. If an MH detects loss of reply, it will send a retrieval request to its previous AB for retrieving the reply after handoff. In our approach, the AB takes the initiative to forward the reply to the MH after handoff.

2.2 Job Queueing and Scheduling

Job queueing model provides a useful tool for predicting the performance of many service systems including computer systems/networks, telecommunication systems, and manufacturing systems [97]. Traditional queueing models predict system performance under the assumption that the service facilities experience failures and repairs.

Gaver [47] analyzed a single server which is subjected to random interruptions and gave the Laplace-Stieltjes Transform (LST) of distributions of a job's completion time under different interruptions: postponable, preemptive-resume, preemptive-repeat-identical, and preemptive-repeat-different. It was stated that interruptions tend to increase the intervals between job departures from the system; however, interruptions may decrease completion time durations. In addition, the author also gave the LSTs and moments of busy period duration, of queue length, and of waiting time.

Nicola *et al.* [95, 97] utilized an irreducible continuous-time Markov chain to model the server's failure and repair behaviors, in which each state is classified as preemptive-resume or preemptive-repeat-identical. They derived the steady state probability distribution and the mean of the number of jobs in the system.

Altiok [8] assumed that the service and repair time distributions are a mixture of generalized Erlang distributions, and no jobs arrive during rollback recovery period, under which the steady state probability distribution of the number of jobs was derived.

All the above models assume that the input to a queue follows a Poisson fashion. In Chapter 4, we conduct queueing analysis for an AB and derive the expected message sojourn time. However, we assume that the message dispatch facility (AB) itself is not subjected to failures, and only the message target, i.e., MH, undergoes failures and handoffs, which cause messages to be blocked and delayed in the message dispatch queue at AB. The behavior of an MH is also modeled with a three-state Markov chain but with different states: normal, handoff, and recovery, in which the handoff state is uniquely provided by wireless mobile environments.

In addition to server breakdowns, the job processing policy in a service facility may also affect the job completion time, thus influencing the number of jobs waiting for processing. Therefore, many papers have been published to address this problem.

Coffman *et al.* [32] derived the LST of the job waiting time distribution for a processor-sharing system, in which the service facility is shared simultaneously by each job in the system. The derived waiting time distribution is conditioned on the service requirement and the number of jobs in the system when a job arrives. Nunez-Queija [98] also gave the LST of the sojourn time distribution in an M/M/1 queue with the processor-sharing service discipline, in which, however, the server is subjected to breakdowns. He pointed out that the expected sojourn time is not proportional to the service requirement.

Rasch [107] derived the distribution of queue size and the average waiting time for a time-shared system using round-robin scheduling, with and without switchover overhead. Each job is assigned with a priority based on its length. In the round-robin scheduling model, the service facility processes each job for a maximum predetermined time period (or quantum). If a job cannot be completed during its assigned quantum, the remainder of the job returns to the end of the single queue for another quantum of service.

Coffman and Kleinrock [33] analyzed two models: the round-robin model and the feedback model. A feedback system contains multiple queues in which a new arrived job joins the tail of the first queue. If the job cannot complete its processing, it joins the tail of the next queue, thus providing rapid service for jobs with short service-time requirement. Kleinrock *et al.* [69] solved the average response time for jobs conditioned on their service requirement, in which three scheduling disciplines are incorporated: first-come-first-served, round-robin, and feedback.

The cyclic polling model has been surveyed in [15, 123, 124], which is a system of multiple queues accessed in cyclic order by a single service facility. Four basic disciplines with respect to the rule by which the service facility leaves a queue exist: In an exhaustive service system, the service facility continues to serve each queue until it empties; In a gated service system, the service facility serves only those jobs that were found in a queue when it visited the queue; In a k-limited service system, each queue is served until either it empties or k jobs are served, whichever occurs first; And in a k-decrementing service system, each queue is served until either it empties or the queue size decreases to k less than that found at the polling instant. After each service completion, the service facility takes a rest before returning for inspection of the queue.

However, most of the aforementioned papers on job processing policy discussed the system performance in the absence of server failures. The points of departure of the work in Chapter 4 from others are that we solve and simulate the message sojourn time at ABs in the presence of failures and handoffs of MHs for wireless mobile environments.

2.3 Program Execution Time

Many analysis results have been derived about the completion time of a program having a finite failure free program execution time executed on a stand-alone host in the presence of failures. On the basis of the derived program completion times with checkpointing, the optimal checkpoint rate have also been investigated.

Duda [39] derived the probability density function of the program execution time and its expectation with and without checkpointing; however, Duda's work assumed that no failures occur during recovery and that the recovery time is a constant. In Chapter 5, we extend the recovery time to be a random variable (r.v.) and allow failures during recovery.

Kulkarni *et al.* [74] carried out the analysis of the program completion time in which checkpointing is allowed during reprocessing of a program, and they utilized the derived results to build a queueing model with a Poisson input. The checkpointing rates to minimize the expected program completion time and response time were derived individually.

Nicola [96] presented the LST and the expectation of the program execution time under different checkpointing strategies: equidistant checkpointing, checkpointing in modular programs, and random checkpointing. The corresponding form of equidistant checkpointing in our message-number-based models is the equi-number checkpointing [25], which is a special case of the deterministic checkpointing described in this thesis. Nicola's selected distribution for random checkpointing was the exponential distribution. As the message number demonstrates a discrete characteristic, we engage the geometric distribution instead of the exponential distribution in our work.

In [39, 74, 96], the authors pointed out that without checkpointing the resulting program execution time is an exponential function of the specified program execution time, and with equidistant checkpointing, the function changes to a linear form. A similar observation with respect to the message number will be made in this thesis.

Young [142] described a first-order approximation to the optimal checkpointing interval which minimizes the time lost due to failures. Garg *et al.* [46] combined checkpointing and rejuvenation and derived the program execution time for the resulting two-dimensional optimization problem of finding the total numbers of equidistant checkpoints and rejuvenations to be performed during the execution of a program.

Gelenbe and Derochette [50] examined the influence of the checkpointing interval in a transaction-oriented database system on the system availability and on the average response time. They employed a three-state Markov chain, including a normal state, a recovery state, and a checkpointing state, to derive the condition for the existence of a stationary probability distribution.

Tantawi and Ruschitzka [125] carried out the analysis of an equi-cost checkpointing strategy, which is failure-dependent yet reprocessing-independent. With the equi-cost strategy, the expected reprocessing time between any two successive checkpoints equals the mean of the checkpointing time. They considered general failure distributions and allowed failures to occur during checkpointing and error recovery. In our model, besides MH failures in checkpointing and recovery, wireless link failures and handoffs are also allowed; however, we restrict the inter-failure time to an exponential distribution.

Chandy *et al.* [21] presented three models through which the optimal checkpointing intervals were derived. In the first model, no errors occur during recovery, and the job arrival rate is constant. The second model takes into consideration possible errors during checkpointing and recovery. The last model considers that the job arrival rate varied with time in a cyclic fashion. They assumed that the recovery time after a failure is proportional to the number of logged jobs since the most recent checkpoint.

Dimitrov *et al.* [37] gave an execution time analysis when checkpoints are placed by a deterministic schedule pre-determined offline, and then investigated an approach for determining the optimal checkpointing schedule.

Gelenbe [49] showed that the optimal checkpointing interval to maximize system availability is a function of the load of the system, and proved that in order to maximize the availability the total operating time of the system between successive checkpoints should be a deterministic quantity.

Grassi *et al.* [51] evaluated the probability distribution of the overhead caused by the usage of the checkpointing rollback recovery technique, through which some checkpointing strategies were developed to minimize the variance of program execution time under an execution time constraint, to guarantee maximum system unavailability, or to minimize the cost introduced by checkpointing.

Krishna *et al.* [70] suggested that the optimization criteria for checkpoint placement should be a trade-off between the benefits derived from checkpointing and the overhead it imposes.

In addition, Ling *et al.* [81] derived an explicit optimal checkpointing frequency formula which minimizes the total expected cost of checkpointing and recovery. L'Ecuyer and Malenfant [76] proposed a dynamic programming approach to find a dynamic decision rule which maximizes the average availability for rollback and recovery at the system level. Plank and Thomason [104] further applied the concept of availability to measure the performance of checkpointing. Ziv and Bruck [144] designed an online checkpoint placement algorithm which keeps track of the state size of a program and looks for points in the program where checkpoint placement is the most beneficial.

All the above analysis results are derived for wired networks. Recently a few investigations have also been conducted for wireless mobile networks. In Page 19, we have stated that Pradhan *et al.* [105] had proposed AB-driven recovery strategies to achieve FT. Furthermore, they reported the expected cost incurred during a handoff period with and without failures. They identified the optimal checkpointing interval, and concluded that the performance of a recovery scheme depends on the mobility of MH and the bandwidth of wireless links. The probability distribution functions of the corresponding recovery times were addressed in [23] as a failure recoverability problem. In Chapter 5, we complement their work with the program execution times with different checkpointing strategies.

2.4 Network Reliability

Two-terminal reliability is a common measure of network reliability, which is characterized by success of at least one path between two specified nodes [2], and is dependent on the topological layout of a communication network in addition to the reliability of individual network components (links and nodes). More general terms are k- and all-terminal reliability. Much work has been done in calculating, estimating, or bounding the reliability of a given wired network. One approach is by employing the combinatorics of network reliability to produce lower and upper bounds with failure-prone links (characterized by link-failure probability), but perfect nodes [38, 78, 140]. The evaluation algorithm proposed in [38] is one of the most computationally efficient method to calculate the two-terminal reliability with perfect nodes [128, 141].

Aggarwal *et al.* [2] developed a concept that the failure of a node implies the failure of links connecting it, with which a symbolic reliability expression derived with the assumption of perfect nodes could be directly modified to incorporate imperfect nodes. Unfortunately, the calculation cost can rise exponentially with the number of links. Torrieri [128] exploited a relation that the event of successful communication over a link is equivalent to the event that both the link and its terminal node are operational. Then he designed an efficient method to compensate for unreliable nodes in network reliability computation. The cost of this method increases linearly with the number of links, and the effect of the unreliable nodes can be directly calculated.

Netes and Filin [93] added the imperfect nodes into paths for decomposing the network into an event-tree, thus considering nodes and links jointly, not separately. Ke and Wang [64] partitioned the network into a set of smaller disjoint subnetworks by only considering links as if all nodes are perfect to directly calculate the network reliability expression instead of using any compensating methods.

Based on previous work, in Chapter 6 we evaluate the reliability of wireless infrastructure network with node failures by the introduced end-to-end mobile reliability. However, we focus on various communication schemes due to MH mobility instead of a static wired network topology. Previous results could be employed into the end-to-end mobile reliability to provide a more detailed and complete reliability assessment for wireless infrastructure network systems.

Nevertheless, the reliability issue in wireless infrastructure networks has been addressed only by a handful of researchers. Reliability and survivability issues of wireless infrastructure networks were discussed in [121], which concluded that each component engaged in the end-to-end connection is a potential point of failure. However, it did not explicitly state how the user mobility, which is unique in wireless infrastructure networks, affects the end-to-end reliability.

Varshney *et al.* [130] modeled and simulated the reliability and survivability of wireless infrastructure networks with a proposed wireless infrastructure building-block (WIB). By scaling the number of WIB, they evaluated network failures and corresponding impacts under various observation durations, component failure characteristics, and network sizes. They pointed out that redundancy and multifunction/multimode devices are two approaches to enhance the network reliability.

2.5 Sleeping Configuration in Wireless Ad Hoc Sensor Networks

2.5.1 Area Coverage

Coverage problems have been formulated in the field of computational geometry as the Art Gallery Problem [115]. A gallery is usually modeled as a simple polygon on a two-dimensional plane. The Art Gallery Problem tries to determine the number and locations of observers, which are necessary to guarantee that every point in the gallery should be monitored by at least one observer [19].

Boolean Sensing Model

Recently several sleeping configuration protocols have been proposed to address the energy conservation and lifetime extension issues in wireless ad hoc sensor networks operating within a coverage constraint. Based on their adopted BSM, they identified redundant sensors by means of geometric computations.

Tian *et al.* [126] proposed an off-duty eligibility rule based on sponsored sector (SS), which considers only the nodes whose distance is less than or equal to the sensing radius. This off-duty rule guarantees complete sensing coverage as long as no void area exists; however, the SS is an underestimation of sensing coverage provided by neighboring nodes and leads to excess energy consumption.

Jiang and Dou [59] improved the work of Tian *et al.*[126] by replacing the communication neighbor with the sensing neighbor, thus utilizing more coverage capability provided by neighbors. Nevertheless, their protocol may not preserve coverage when sensors have different sensing radii.

Yan *et al.* [136] introduced a differentiated surveillance service for sensor networks. In their proposed service, a sensor first calculates a time reference point and the corresponding time duration for each covered grid sampling point; then the sensor unites all the time durations and takes the result as its working time. The authors additionally provided FT by periodically broadcasting a heartbeat message; however, utilizing the heartbeat message to detect sensor failures is too energy-expensive in sensor networks. In our work, FT is an intrinsic capability in the CSM.

Huang *et al.* [58] formulated the coverage problem as a decision problem. Whether a sensor is eligible to sleep is determined by observing how the perimeter of its sensing range is covered by its neighbors. But they did not give detailed sleeping configuration protocols for distributed sensors.

Ye *et al.* [139] developed PEAS, a probing mechanism designed to conserve energy. The PEAS does not require any node to maintain a knowledge of the states of its neighboring nodes and it distributes node wake-ups randomly over time. When a sleeping sensor wakes up, it detects whether any working sensor is present within a certain probing range by broadcasting a probing message and waiting for a reply. If no reply is received within a time period, it starts working until it fails or depletes all its energy. In this solution, the application specified probing range indirectly determines the degree of coverage. However, this probing-based approach has no guarantee of adequate sensing coverage.

Slijepcevic *et al.* [120] described a heuristic algorithm to select mutually exclusive groups of sensors, where members of each of those groups together cover the interested area completely. Only one such group is working at any moment. After a specified interval, another group is activated. By maximizing the cardinality of the group set, the proposed algorithm achieves energy savings while fully preserving coverage.

All the aforementioned algorithms assume that each node is aware of its own location. By relaxing the requirement of completely preserving area coverage, Tian *et al.* [127] presented three location-free sensor scheduling rules: nearest-neighbor-based, neighbor-number-based, and probability-based, which compare the distance between a sensor and its nearest-neighbor, the number of a sensor's neighbors, and a randomly generated number, with a predefined corresponding threshold, respectively. We also present a sensor sleeping eligibility condition which estimates the area coverage loss. It still needs node location information; however, it has the advantage of being able to identify critical regions in the deployed area.

Collaborative Sensing Model

The CSM has also been exploited in the literature. Megerian *et al.* [90] defined exposure as an integral measure of how well a sensor network can observe an object, moving on an arbitrary path, over a period of time.

Xing *et al.* [134] exploited a probabilistic distributed detection model with a Coordinating Grid (Co-Grid) protocol. Utilizing the concept of data fusion and the majority rule, the fusion center located at the center of each grid decides which sensors in its fusion group should be in active status and dispatches its decision to its group members. Then the group members set their status accordingly. The Co-Grid engages overlapping grids to reduce the boundary effect that exists when there is no inter-grid coordination between two adjacent non-overlapping grids. However, this approach is not fully localized, and fixing the location of the fusion center is not practical when sensors are randomly deployed. Our approach allows each sensor to decide its status locally, thus there is no centralized decision. Each sensor can function as a data fusion center for its neighboring sensors, and no stringent grid is imposed. These two characteristics simplify the construction of cluster-based protocols.

Liu and Towsley [82] studied the coverage problem from a theoretical perspective with three measures: area coverage, node coverage, and detectability. Both sensing models, the BSM and the CSM, were evaluated in their paper, and some asymptotic characteristics were developed.

2.5.2 Network Connectivity

To address the problem of providing communication connectivity within an energy conservation context, topology control protocols have been proposed that keep awake only the nodes necessary to maintain continuous routing.

SPAN [22] is a randomized algorithm in which nodes make local decisions on whether to sleep or to join a forwarding backbone network as a coordinator. The coordinators stay awake continuously and perform multi-hop packet routing, while other nodes remain in sleeping mode and periodically check if they should wake up and become a coordinator.

GAF [135] employs geographic location information to partition a monitored area into virtual square grids and considers all nodes in a particular grid square to be equivalent with respect to forwarding packets. It performs leader rotation among the nodes inside the virtual grid in order to balance energy consumption.

LEACH [55] divides a network into clusters and randomly rotates the cluster leader in order to distribute the energy consumption evenly among the sensors. A sensor decides to which cluster-head it belongs by evaluating its minimum communication energy needed to connect with the cluster-head.

In ASCENT [20], each node adapts its participation in the multi-hop network topology based on a measurement of local connectivity and packet loss information; therefore, no location information is needed.

2.5.3 Area Coverage and Network Connectivity

Furthermore, some studies have been published on maintaining sensing coverage and communication connectivity together. Wang *et al.* [133] and Zhang *et al.* [143] both proved that without location error, the sensor's communication radius being at least twice of the sensing radius is the sufficient condition to ensure that the complete 1-coverage of a convex area implies network connectivity.

Wang *et al.* [133] designed an approach called Coverage Configuration Protocol (CCP), which employs the concept that the coverage degree of intersection points of sensing perimeters indicates that of a convex region. However, it is not valid with inhomogeneous sensors with different sensing radii. The protocol developed by Zhang *et al.* [143] is called Optimal Geographical Density Control (OGDC), based on a theorem extracted from [52] pages 59 and 181. In the OGDC, a sensor is randomly selected as the starting sensor in each scheduling round, then other sensors, which are as close to optimal locations as possible, are selected to be working sensors. Their approaches are built on top of the BSM, and both approaches assume that each sensor knows its accurate position; however, this assumption is relaxed in this thesis.

Zou and Chakrabarty [145] proposed a coverage-centric active nodes selection (CCANS) algorithm to maintain coverage and connectivity together based on a probabilistic sensing model and constrained their evaluation within one-hop neighbors. The CCANS is a token-based protocol by assigning the token to exactly one sensor at any time instant, in which only the sensor with the token evaluates its sleeping eligibility. In this thesis, we integrate the advantage of one-hop communication into the coverage problem directly by employing the neighboring-sensor field sensibility. Furthermore, a sensor is a sleeping candidate when its sleeping does not reduce the original coverage and does not break its one-hop neighbors into communication partitions.

2.5.4 Voronoi Diagram

Voronoi diagram [11, 101] has also been investigated in the coverage problem of sensor networks. Based on Voronoi diagram, Meguerdichian *et al.* [89, 91] formulated the coverage problem with maximal breach and maximal support paths to determine the best- and worst-case coverage for agents movement. The best-case coverage problem was solved by Li *et al.* [79] with efficient distributed algorithms. However, the coverage definition in these two papers is somewhat different with ours. Carbunar *et al.* [17, 18] utilized Voronoi diagram to detect the coverage boundary, which is a necessary condition in our developed criterion for selecting sleeping-eligible sensors with the BSM. Wang *et al.* [132] developed a sensor movement strategy by adjusting the distance between a sensor and its farthest Voronoi vertex to maximize the sensor coverage; however, our work focuses on static sensors.

□ **End of chapter.**

Chapter 3

Message Logging and Recovery in Wireless CORBA

The emerging mobile wireless environment poses exciting challenges for distributed FT computing. This chapter proposes a message logging and recovery protocol on the top of wireless CORBA and FT-CORBA architectures. It employs the storage available at the AB as the stable storage to log messages and checkpoints on behalf of MHs. Our approach engages both the quasi-sender-based and the receiver-based logging methods and makes seamless handoff in the presence of failures. The details of how to tolerate MH disconnection, MH crash, and AB crash are described. The normalized execution time of an MH engaging our proposed scheme and the handoff effect are evaluated.

3.1 Fault Tolerance Model

Figure 3.1 presents an architectural overview of our FT model. Our approach is based on message logging and checkpointing. The message logging mechanism in ABs applies different methods for messages received from and sent to TBs. An AB logs messages after it receives them from an TB (receiver-based), but logs messages which are received from other hosts (ABs or SHs) before it sends them to the TB (quasi-sender-based). The TB may send back an acknowledgement depending on the received message's type. We checkpoint an MH's state periodically by a local timer or when the amount of the received messages exceeds a predefined threshold.

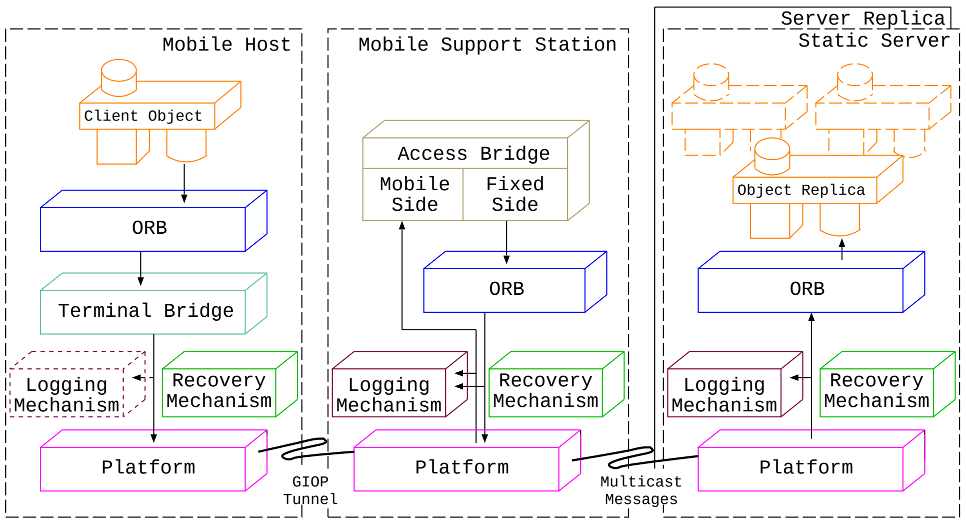


Figure 3.1: Fault-tolerant wireless CORBA architecture.

An MH may become unavailable due to (i) MH disconnection, (ii) MH crash, and (iii) AB crash. As mentioned in Subsection 1.1.2, physical damage becomes more probable to an MH than to an SH. It is limited to lower processing power, lower memory resources, and lower power supply. It can be disconnected from network intended or unintended. It may often move from one AB to another. So an MH is not suitable to act as stable storage. But when it is disconnected and the user still wants to operate continuously using local information, the FT protocol may save checkpoints in its local disk in order to recover from some transient faults, such as operating system crash and battery discharge [94], etc. So we depict the logging mechanism in MH with dashed lines.

An AB is on the border between wireless and wired networks. In the wireless CORBA architecture, all messages to and from an MH are traversed through ABs. Every message leaves a local copy in AB automatically. Therefore, it does not need to send an extra copy of each message elsewhere for logging purpose to tolerate MH crash. So we choose the storage at AB as stable storage for the message logging and checkpointing protocol. But the mobile computing environment does not restrict a user's location. When a user moves from one AB to another, the carried MH should change its connected AB. So the location of the stable storage would also be changed during handoff [105]. It is one of the duties of our recovery protocol to find where the last checkpoint is located. Finally, an AB contains multiple associated MHs at the same time, but these MHs utilizes the AB only as a gateway for transforming and forwarding messages, and there is no dependency between these MHs from the viewpoint of the AB . So an AB keeps different logs for its different associated MHs.

The SH and HLA are replicated in *passive* or *active* style according to the FT-CORBA standard (Chapter 23 in [99]). An AB communicates with SHs and HLAs by a group communication system, which should detect and suppress duplicate requests and replies, and deliver a single request or reply to the AB. So there is no single point of failures in our architecture. In this chapter, we do not discuss this FT strategy in detail as it has already specified by the FT-CORBA standard.

3.1.1 Data Structures

We employ the following data structures in our message logging and checkpointing mechanism.

- *Sequence Number (SN).* Each message exchanged in the GIOP Tunnel between an AB and a TB has an SN, which identifies the message itself and the order in which the message is sent. An AB ensures that the SN is distinct for a dedicated TB, but the SNs may be same among different TBs.

- *Message Log Entry (MLE).* Two types of MLEs are employed for two message logging mechanisms (receiver-based and quasi-sender-based). The first type contains a message received by an AB from a TB and the status after the AB processes it. The second one is for messages sent to a TB, which includes an additional SN of the corresponding acknowledgement message. The additional SN indicates the order in which the message is received by the TB.

- *CheckpointData* and *CheckpointDataReply* Messages. When an MH takes a checkpoint, it utilizes these two messages to reliably save the checkpoint in the current AB.

- *PurgeCheckpoint* Message. This message is sent out to clear old checkpoints when an AB receives a *CheckpointData* Message. It needs not be delivered reliably.

- *FetchCheckpoint* and *FetchCheckpointReply* Messages. A *FetchCheckpoint* Message is initialized when an MH detects a failure and starts to restore the state before the failure. A *FetchCheckpointReply* contains the last checkpoint of the MH.

3.1.2 Message Logging and Checkpointing

The steps in message logging, illustrated in Figure 3.2, are (1) An MH sends a request message x via a GIOP tunnel to the currently connected AB; (2) The AB logs x in its local stable storage pessimistically, sends an acknowledgement back to the MH, and relays x to the remote static server, then the AB waits for a reply; (3) After receiving the reply message y, the AB logs y and dispatches it to the MH; (4) The MH sends a message back to acknowledge y and delivers y to the top level; (5) The AB logs the SN in the acknowledgement message with message y.

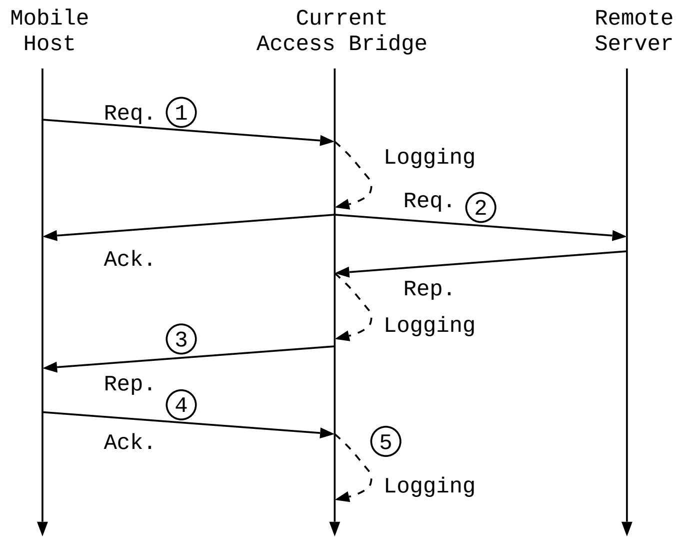


Figure 3.2: Normal invocation sequence with message logging.

When a local timer in the MH expires (time-based checkpointing), or the MH receives a predefined number of messages since its last checkpoint (deterministic or random checkpointing), the MH will initialize a checkpointing procedure. The hosted TB encapsulates the checkpoint in a *CheckpointData* message and sends it to the current AB. To save wireless bandwidth, the checkpoint may not be sent out immediately, and it can be piggybacked with the next message from the TB to the AB [103]. The AB logs the received checkpoint on its local stable storage and informs the HLA that a new checkpoint of the MH is saved in this AB. This information will be utilized in the recovery process to fetch the last checkpoint. The checkpointing interval is determined by the application requirements and the failure rate of the MH.

It is also determined by the handoff frequency of the MH. The effect of different checkpointing strategies and checkpointing intervals will be analyzed in Chapter 5.

If an MH saves a checkpoint in its currently associated AB, all the message logs and checkpoints prior to this checkpoint can be deleted since they are no longer necessary for recovery of this MH. So the AB will send a *PurgeCheckpoint* message to the HLA to delete those obsolete checkpoints and MLEs. This message needs not be reliably delivered, so long as any future *PurgeCheckpoint* message for the same MH will be delivered [61]. After the HLA receives the purge message, it will forward this request to the ABs in the itinerary track of the MH so that they can purge the unnecessary checkpoints and messages and reclaim the stable storage. No MHs and wireless communications are involved during storage reclamation.

3.1.3 Mobile Host Handoff

In wireless networks which are organized in communication cells, a handoff is a mechanism for an MH to seamlessly change a connection from one AB to another. Handoff can be started due to two causes: normal operation and sudden connectivity loss. In the normal operation, the MH will create a connection with a new AB. But in the second case, there is another successful outcome of the handoff procedure: connectivity re-established to the same AB as before [100]. Therefore, we identify two ABs in a handoff procedure with:

- Old Access Bridge (OAB) that was connected by an MH before the handoff.
- New Access Bridge (NAB) that would be connected after the handoff, which may be the same as the OAB.

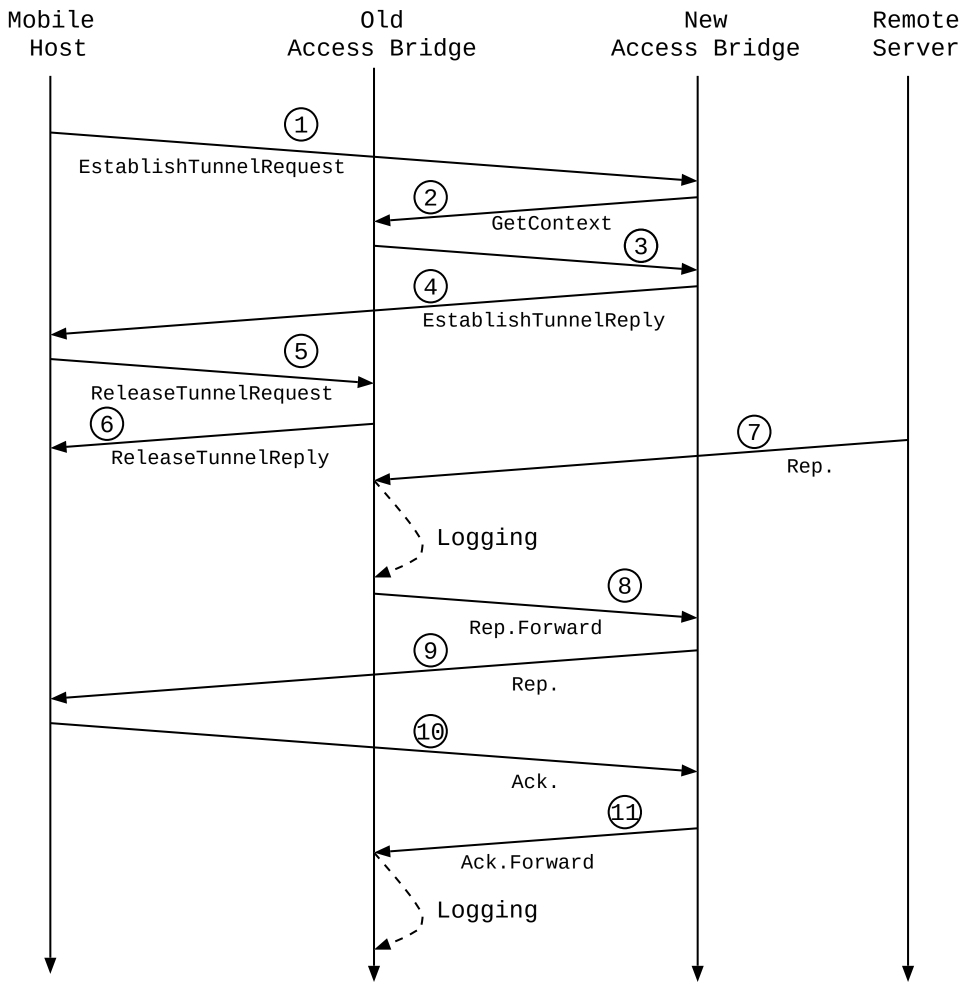


Figure 3.3: Handoff procedure.

Figure 3.3 depicts the handoff procedure where an MH reestablishes communication connectivity to a new but different AB. First, the MH creates a network connectivity (in network layer) with the NAB. Then it sends a request message to establish a tunnel (message 1). The NAB uses information contained in the request message or acquired by querying the HLA to obtain the OAB of this MH. The NAB sends a message to the HLA to update this MH's location and invokes a handoff operation at the OAB (message 2). The

OAB forwards necessary context data, such as *Sequence Number, Last Sequence Number Received, Connection ID*, to reconstruct the execution context in the NAB (message 3). The NAB sends the tunnel establishment reply to the MH (message 4) and the MH breaks the connection with the OAB (message 5 and 6). Afterwards, the MH sends and receives all messages through the NAB. The messages received by the OAB during the handoff (message 7), such as replies to former requests, etc., are forwarded to the NAB (message 8), and the NAB relays them to the MH (message 9). The acknowledgement messages (message 10) are forwarded to the OAB (message 11) to update the corresponding message status in the logs to keep the integrity of these MLEs. The GIOP requires that a reply should be sent in the same GIOP connection as the request came in [100]. So if a message is a reply for a request that was received through the OAB before the handoff, this message is encapsulated in a forward format, and when the NAB receives it, the NAB should relay it to the OAB. All these forwarded messages are logged in the OAB.

3.1.4 Mobile Host Disconnection

An AB functions as a proxy between an MH and other hosts (SHs and ABs). Its major function is to forward messages to and from MHs. We construct an AB with two parts (see Figure 3.4): mobile side and fixed side [100]. The mobile side connects with an MH by a GIOP Tunnel, while the fixed side creates normal IIOP (Internet Inter-ORB Protocol) connections to communicate with remote hosts. The AB keeps different maps between these two parts by the *Connection ID* specified in [100] for every associated TB. Using the AB as a proxy, we can hide sudden MH disconnection from remote hosts [137].

According to the fact that an AB is a proxy for relaying GIOP messages, we define three status of a message in an AB.

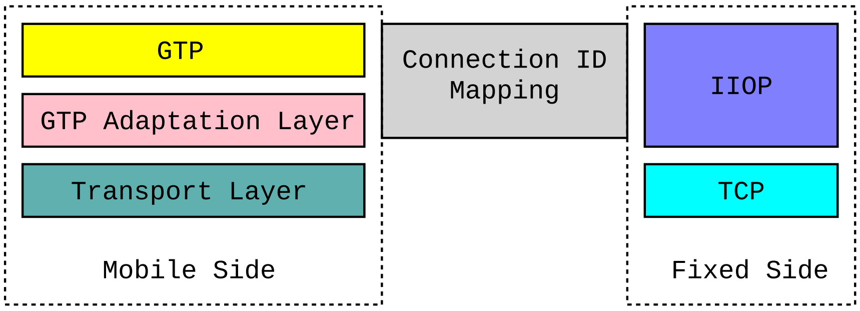


Figure 3.4: Access Bridge ORB.

- *Received.* This is the default status for a message when the AB receives this message.

- *Sent.* When a message is relayed to an MH, an SH, or another AB but before receiving an acknowledgement or a reply, the status of the message is *Sent*.

- *Processed.* After the AB receives an acknowledgement message or a reply, it changes the corresponding message's status to *Processed*.

If the AB receives a message which does not need to be relayed, the status of this message will be directly changed to *Processed* after the AB processes this message.

During a sudden MH disconnection, the last connected AB still keeps IIOP connections with remote hosts for a predefined time period. When the AB receives messages from the remote hosts, it logs messages but does not forward them to the target MH (as message 7 in Figure 3.3). When the AB receives a notification that the MH reconnects with the network, it forwards these received-but-not-sent messages to the MH (reconnects with the same AB) or the MH's currently associated AB (reconnects with a different AB). If the MH recovers the connection with the same AB in a predefined time period, the AB will reuse these IIOP connections for subsequent communications. Otherwise, the AB will terminate all these IIOP connections established for this MH. If the AB has received all the reply messages (there are no outstanding request messages), it closes these connections intermediately.

3.1.5 Mobile Host Crash

During disconnection, the state of an MH is kept intact. However, in the case of MH crash, its local state is lost and needs to be recovered from a checkpoint and subsequent message logs. The MH is assumed to be fail-stop [14], i.e., the associated AB is able to detect the failure of the MH. Each failed MH can perform handoff and recovery procedure independently, which means that no other MHs need to roll back together.

First the MH initiates a handoff procedure as depicted in Subsection 3.1.3. After a successful handoff, it starts a state recovery procedure, which includes four phases:

1. The HLA finds the location of the last checkpoint and forwards it to the NAB;

2. The HLA collects all successive MLEs from the itinerary track of the MH and forwards them to the NAB;

3. The NAB sends the checkpoint, sorts the *processed* messages by their corresponding acknowledgement SNs, forwards these messages sequentially, and delivers the *sent* messages sequentially in their own SN order;

4. The MH initializes the application using the checkpoint and then executes the application. If a generated message has a counterpart message in the MLE set, this message is inhibited and will not be sent out.

After applying the recovery procedure, the state of the MH will be restored to the state just before the failure. (We assume that the application is deterministic.)

A GIOP tunnel is shared by all GIOP connections to and from the TB [100], so some messages maybe arrive at the TB earlier than the messages sent before them. We adopt a quasi-sender-based message logging mechanism for these messages [61], which means that the AB acts as a message sender from the viewpoint of the TB. For reconstructing the same sequence of messages arriving before failures, we employ the SNs of the corresponding acknowledgements in MLEs to sort these processed messages before sending them out sequentially.

In our approach, if a user moves from one AB to another, the stable storage for storing checkpoints and messages is changed accordingly. So if the mobile user traverses many times during a checkpointing period, the logged messages are scattered in these ABs. If we want to recover an MH from a failure or to revoke the stable storage for outdated messages, we need a method to find all these messages. Because the HLAs keep one itinerary track for each MH, we can employ the tracks to facilitate the messages collection and storage revocation, as described in Section 3.1.2.

The recovery period is time consuming because visiting different ABs is required to collect necessary MLEs. The reason is that when a failure occurs, the messages are scattered. We can improve this recovery procedure by collecting messages to a stable storage near or in the current AB. A strategy proposed in [105] ensures that the message logs and the checkpoint corresponding to the MH are at the "predecessor" AB. To achieve this, during handoff, a message is sent to the predecessor AB to transfer the checkpoint and logs. But if the MH moves frequently to another AB, this strategy will still create heavy volume of data transfer. We improve this strategy by letting the HLA trigger the transfer of the checkpoint and logs. In the HLA, there is a daemon and an array of timers for each MH. If one timer is expired, the daemon will dispatch a thread to handle the data transfer for the corresponding MH. The thread will collect the last checkpoint and successive message logs and save the data in the current AB of this MH. The timer is adaptive. It will extend the time period if an MH moves frequently, and it will shorten the time period if the MH maintains connection with an AB for a long time. If a checkpoint is taken during this message collection period, the HLA stops the related thread. We also can use the number of handoffs or the distance between the currently associated AB and the AB which contains the last checkpoint as the trigger of message collection.

3.1.6 Access Bridge Crash

An AB facilitates the connection mapping between an MH and an SH, so the AB is in the critical path. For tolerating AB crash, normal replication strategies can be adopted [5]. Recognizing the nomadic feature and the handoff mechanism in the mobile computing environment, we utilize a strategy that replicates the execution context and messages in an AB to its *previous* AB for each MH. A *previous* AB for an MH is an AB in its movement track just one hop before its current AB. If there is no movement track for this MH, we choose the HLA as the "previous" AB. This replication strategy is passive. Some messages that do not change the status of the AB will not be replicated. Because each MH has different movement tracks, this strategy generates different AB replicas for different MHs. After an AB failure, different MHs can move to different NABs to start the handoff procedures.

If an AB crashes, the MH will detect this failure and then start a handoff procedure. If there is only one AB covering the current location area, the mobile user should explicitly move to another location for handoff. The handoff procedure has some differences from the normal handoff. The NAB first queries the HLA for the location of the replicated message logs and makes a request to the AB. The AB reconstructs the execution context from the message logs and sends this context to the NAB. The NAB initializes a new context for this MH according the received context and resends those messages which have no acknowledgments or replies and whose SNs are not in the vector which contains all the SNs of the messages received by this MH after its last checkpoint. After a successful handoff, the NAB informs the HLA that the recovery procedure is finished, and the MH continues to work as in normal condition. The HLA removes the failed AB from the MH track to avoid to select the failed AB as a previous AB. If the MH moves back to the previous AB, the recovery procedure will be more efficient because all messages required to recovery are in the local storage. If the AB restarts after a failure, the MH can create connectivity with this AB just as a normal handoff from the previous AB.

To avoid re-executing resent messages after an AB crash, a remote server should do some special work. When the server sends a reply message to the crashed AB, it learns that the AB is not reachable and then logs this reply message locally. After a successful handoff, the NAB reissues the same request through a new GIOP connection, the server identifies this request, retrieves the corresponding reply from its local log, and sends it back. Therefore, the server processes the same request only once and keeps the data consistent.

3.2 Simulations and Evaluation

A simulation model is constructed to evaluate our proposed scheme, which consists of 1 MH, 5 ABs, and 2 SHs. The MH sends request messages to SHs which are selected randomly, and the time interval between two successive messages is exponentially distributed with a mean of 0.1. The MH moves around in the mobile computing environment with a handoff rate which follows a Poisson process with rate ρ. Each AB has a static route to the SHs. The failure rate of the MH follows a Poisson process with rate 0.001[1] . We assume that a failure is detected as its occurrence, so an MH performs recovery procedure instantly after a failure. The service rate of an SH is 0.1. Let the ratio of the average cost transferring an application message or a checkpoint to the average cost transferring a control message over one hop of the wired network be 10, and the ratio of the cost transferring a control message over one hop of the wireless network to the cost over one hop of the wired network be also 10. The checkpointing rate is τ. Every request has a corresponding reply, and the MH completes successfully if it receives all the reply messages.

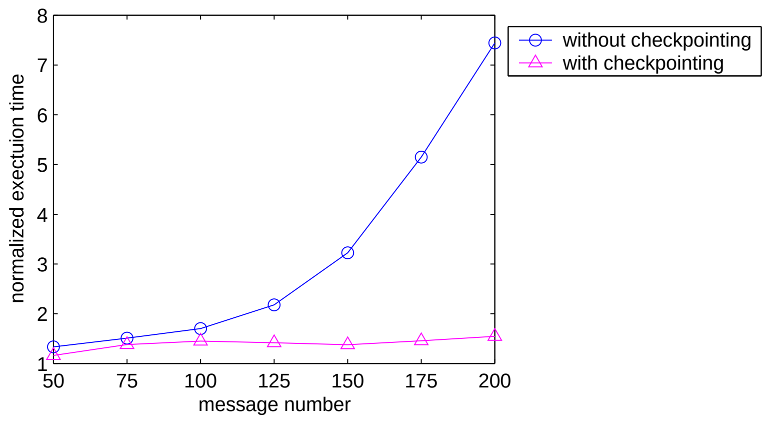


Figure 3.5: Program execution time with and without checkpointing.

Figure 3.5 shows the execution time of the MH engaging checkpointing or not. In this measurement, no handoff occurs. The execution time is normalized to the execution time without failures and handoffs. From this figure, we know that without checkpointing the normalized execution time increases dramatically as the message number increases. (We will observe that this relationship is exponential in Chapter 5). After engaging checkpointing, the time

[1]The unit of all failure and service parameters is $1/s$ (one per second) in this thesis.

curve is plotted nearly horizontally. So the execution time with checkpointing has a linear relationship with the message number. We also can see that checkpointing and message logging incurs overheads due to that no application message can be sent out during checkpointing and the message logging mechanism delays the message delivery.

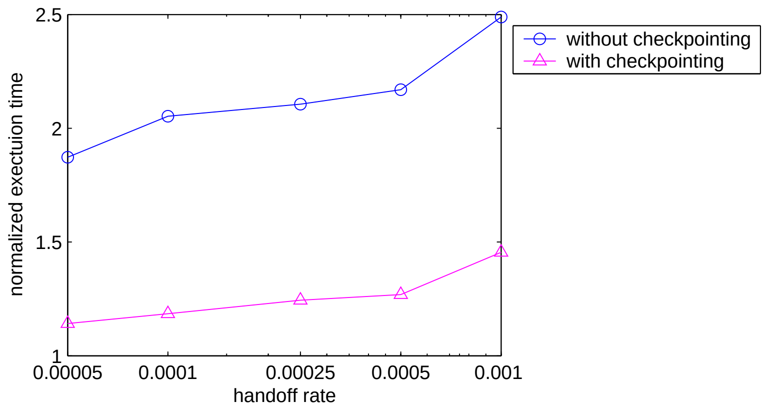


Figure 3.6: Program execution time vs. handoff.

To demonstrate how the handoff influences the execution time, we let the MH moves randomly in the ABs. In this simulation, the same parameters values are utilized, and the message number is 100. Figure 3.6 shows the results which are also normalized to the execution time without failures and handoffs. It implies that the execution time increases linearly as the handoff rate increases. As we described, no application message can be transmitted to or from the MH during handoff. After a handoff, the OAB has to forward its received replies during the handoff to the NAB. When a failure occurs, the checkpoints and message logs may be scattered in several ABs. All these increase the total execution time.

3.3 Summary

This chapter describes a message logging and failure recovery protocol in wireless CORBA. It employs both quasi-sender-based and receiver-based message logging methods. The protocol can tolerate MH disconnection, MH crash, and AB crash. It chooses the storage available at AB as stable storage to log messages and checkpoints. To tolerate AB crash, it replicates an AB's state on the previous AB for each MH. It also engages the handoff mechanism as an approach to recover from AB crash. Note that the differences of message logging and recovery protocols between wireless and wired networks are introduced by the fact that wireless networks engage handoff operations and the fact that MH is not stable storage. If messages are logged in the core network, the message collection procedure caused by handoff could be removed. However, other issues, such as MH disconnection, AB crash, etc., still exist. In addition, more failure-free overload may be introduced as stable storage is far away from MH.

A simulation model is constructed to evaluate the proposed scheme. After engaging checkpointing and message logging, the program execution time increases linearly as the message number increases in the presence of failures. The handoff affects the execution time by delaying message delivery and by scattering checkpoints and message logs in multiple ABs.

□ **End of chapter.**

Chapter 4

Message Queueing and Scheduling at Access Bridge

The previous chapter shows that AB plays an essential role in FT architectures for mobile computing environments which engage wireless networks. It is the performance bottleneck in the presence of failures and handoffs of MHs. Different message dispatch strategies impose various effects on the message sojourn time at AB. In this chapter, we study five dispatch models: the basic queueing model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model [27]. We derive the expected message sojourn times at AB under steady state for the static and the dynamic processor-sharing models, and simulate all the five models with C-Sim [62]. We observe that the basic model and the static processor-sharing model demonstrate the worst performance. The other three models cut down the sojourn time by dynamically reducing the probability of message blocking which is introduced by failures and handoffs of MHs; however, which one is the best dispatch strategy depends on the specific environments. These analysis and simulation results can help designers of wireless networks explore better FT features of mobile systems for their performance and reliability.

4.1 Mobile Host's State Transition

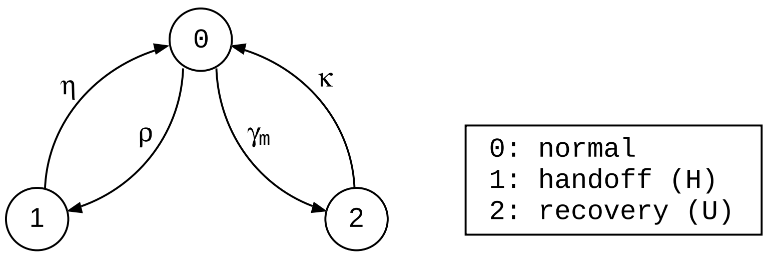


Figure 4.1: MH's state transition.

An MH undergoes handoffs during its application execution and conducts recoveries after failures; therefore, it experiences three states: the normal (operational) state, the handoff state, and the recovery state. Let $\Lambda(t)$ be the state of the MH at time t. $\{\Lambda(t), t \geq 0\}$ is a three-state Markov process with a state-transition diagram shown in Figure 4.1. State 0 is the normal state, during which computational messages can be dispatched to the MH. If a handoff occurs, the MH transits from state 0 to state 1. The time between two successive handoff events is modelled as an exponentially distributed r.v. with parameter ρ. The handoff completion time H is also an exponentially distributed r.v. but with parameter η. The MH may fail and then enters state 2. We assume that the instants of the occurrences of failures to an MH form a homogeneous Poisson process with parameter γ_m and that a failure is detected as soon as it occurs. A recovery process will be conducted in state 2. The recovery time U is regarded as an exponentially distributed r.v. with parameter κ. For simplicity, no failures take place during the handoff or recovery period. According to the assumptions made before, we get a rate matrix of the three-state Markov process $\{\Lambda(t), t \geq 0\}$ for the MH, which is

$$Q = \begin{bmatrix} -\rho-\gamma_m & \rho & \gamma_m \\ \eta & -\eta & 0 \\ \kappa & 0 & -\kappa \end{bmatrix}. \tag{4.1}$$

The stationary distribution of the Markov process is given by

$$\begin{bmatrix} p_0 & p_1 & p_2 \end{bmatrix} = \begin{bmatrix} \frac{\eta\kappa}{\eta\kappa+\kappa\rho+\eta\gamma_m} & \frac{\kappa\rho}{\eta\kappa+\kappa\rho+\eta\gamma_m} & \frac{\eta\gamma_m}{\eta\kappa+\kappa\rho+\eta\gamma_m} \end{bmatrix}, \tag{4.2}$$

in which p_i, $i = 0, 1, 2$, denotes the probability of an MH in state i in the stationary situation. Due to the fact that time intervals between failures and handoffs are relatively large in general, it is usually assumed that the MH will be in steady state most of the time between successive failures and handoffs [54].

4.2 Message Sojourn Time

The message arrival process for each MH used throughout this chapter is assumed as a Poisson process with parameter λ, i.e., the arrival intensity. If the number of MHs covered by an AB is $m > 0$, the total message arrival rate will be $m\lambda$ for an AB since the combination of Poisson inputs is still a Poisson input. The message dispatch requirement is assumed to be an exponentially distributed r.v. D with parameter μ if the dispatch facility of an AB is occupied by one message exclusively, which means that the service rate of the dispatch facility is the constant μ. The *message sojourn time* at an AB, denoted as T, is the duration of a period between the instant when the message enters the AB and the instant when the message is totally disseminated. If a message starts its dissemination, the target MH should stay in its normal state until the current message's dispatch is completed. This assumption is reasonable as the message dispatch requirement is relatively small under common situations. No messages can be distributed to an MH if the MH is in the handoff or recovery state; therefore, an MH in these two states is called as an unavailable MH. The traffic intensity ρ_a for an AB is defined as $\rho_a = m\lambda E(D) = m\lambda/\mu$. With various dispatch strategies, the sojourn times are different in the presence of failures and handoffs of MHs. In the following subsections we will consider five dispatch strategies: the basic model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model.

4.2.1 Basic Dispatch Model

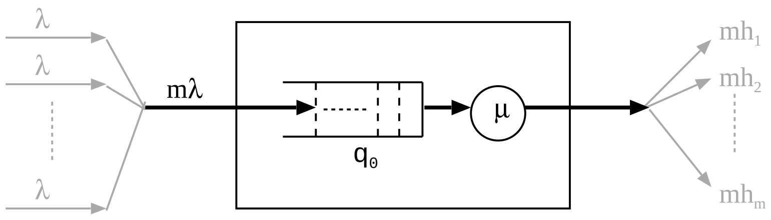


Figure 4.2: Basic dispatch model.

In the basic dispatch model, all messages for various MHs share one queue and are served with the first-come-first-served (FCFS) discipline, as shown in Figure 4.2. We know that without handoffs and failures of MHs, the queue is an ordinary M/M/1 queue that the expected message sojourn time $E(T_{ba})$ is $1/\mu(1-\rho_a)$ [53, 67]. However, with handoffs and failures of MHs, the basic dispatch model evolves as an M/G/1 queue. A message is deliverable when it is at the head of the queue and its corresponding MH is in the normal state. When a message is going to be dispatched, the dispatch time is D. Otherwise, if the message is at the head of the queue and its targeted MH is in the handoff or recovery state, it should be blocked until the MH returns back to the normal state. We assume that when the MH is available, its blocked messages can be delivered instantly. With the memoryless property of the exponential distribution, the residual handoff and recovery times are still H and U, respectively, perceived by a message entering the dispatch facility. When the queue is not empty and there is only one MH, i.e., $m = 1$, every handoff or failure will impose messages to be blocked. However, when $m > 1$, an MH's handoffs or failures may not impose any effects on message dispatch as long as its messages are not at the head of the queue. Therefore, those handoffs and recoveries do not block messages to be dispatched to other MHs. This feature complicates the analysis of a message's waiting time in the queue.

4.2.2 Static Processor-Sharing Dispatch Model

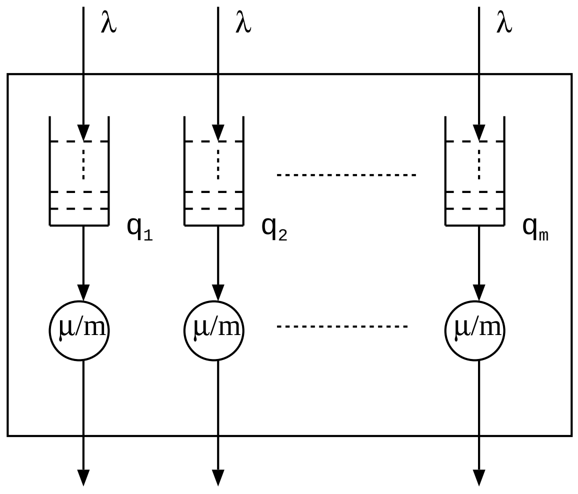


Figure 4.3: Static processor-sharing dispatch model.

In the basic dispatch model, when the message at the head of the queue is blocked, all subsequent messages will be blocked even though they could be dispatched if they get a chance to enter the dispatch facility. To solve this problem, we construct different queues for various MHs and come to the static processor-sharing model, as shown in Figure 4.3. In this model, messages arriving at an AB will be diverted to one of m queues according to their target MHs. The dispatch facility is equally and statically shared by these m queues, which implies that each queue is virtually associated with a dispatch facility whose service rate is μ/m. Only messages targeted at MH i will be filled in queue q_i. The expected message sojourn time could be analyzed using a priority queueing model with the head-of-the-line priority [68], as shown in Figure 4.4.

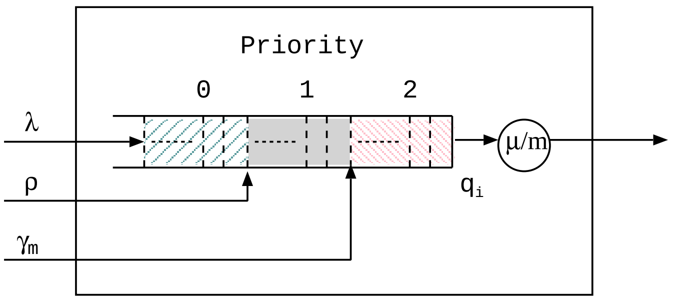


Figure 4.4: Head-of-the-line priority queue for static processor-sharing dispatch model.

As we have assumed that if a message starts its dissemination, the target MH should stay in its normal state until the completion of the message dispatch. This assumption indicates that the priority model is nonpreemptive. According to the fact that during an MH's handoffs and failures, no messages are served, we model the MH's handoffs and failures as special messages with priority 1 and 2, which require service times H and U, respectively. The original messages are treated as priority 0. The larger the value of the priority is, the higher the priority is; that is, messages from priority p_o are given preferential treatment over messages from priority $(p_o - 1)$. Given higher priorities to handoff and failure events, a normal message will only be dispatched when all its previous handoffs and failures have been processed, and those events occurred when it is waiting in the queue.

The expected message sojourn time can be directly derived from (3.31) in [68], which is

$$E(T_{sps}) = \frac{m}{\mu} + \frac{\frac{m^2\lambda}{\mu^2} + \frac{\rho}{\eta^2} + \frac{\gamma_m}{\kappa^2}}{(1 - \frac{m\lambda}{\mu} - \frac{\rho}{\eta} - \frac{\gamma_m}{\kappa})(1 - \frac{\rho}{\eta} - \frac{\gamma_m}{\kappa})}. \tag{4.3}$$

The first term in this expression is the average dispatch time, and the second term is the average waiting time, whose numerator stands for the mean residual life of a message dispatch time, a handoff completion time, and a recovery completion time as observed by a message arrival. The effect of messages, handoffs, and failures present in the queue when a message arrives is given by the first term in the denominator, and the effect of handoffs and failures arriving during the message's queueing time is given by the second term in the denominator.

In the case of the static processor-sharing model, some capacities of the dispatch facility might be wasted when certain queues are empty or contain undeliverable messages, which means that allocating the dispatch facility only among queues which contain deliverable messages may improve the performance. This introduces the dynamic processor-sharing dispatch model.

4.2.3 Dynamic Processor-Sharing Dispatch Model

The dynamic processor-sharing dispatch model puts messages into m queues the same way as the static processor-sharing dispatch model does; however, the dispatch facility is dynamically shared only among the MHs who are in the normal state and whose corresponding queues are not empty. This condition can be restated that the dispatch facility is dynamically shared among queues which contain deliverable messages. Figure 4.5 shows this dispatch model. At

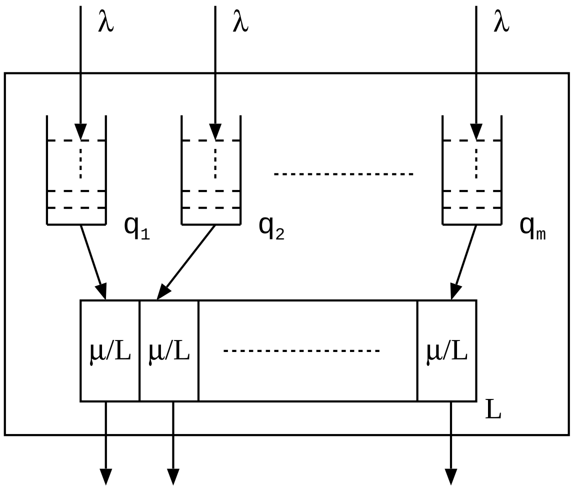


Figure 4.5: Dynamic processor-sharing dispatch model.

a specific time, the dispatch facility contains L deliverable messages, in which each message comes from different queues. As the behaviors of m queues are the same, without loss of generality, we only analyze the experience of a message at the head of q_i. When the message becomes deliverable and enters the dispatch facility, there are other $(L-1)$ deliverable messages in the dispatch facility. Therefore, this message receives dispatch service with rate μ/L, and the traffic intensity varies with L. Let ρ_d be the expected traffic intensity for each queue, then

$$\rho_d = \lambda E(L)/\mu. \tag{4.4}$$

Since

$$Pr(L=l) = \binom{m-1}{l-1} (\rho_d)^{l-1}(1-\rho_d)^{m-l},\ 1 \le l \le m,$$

thus,

$$E(L) = 1 + (m-1)\rho_d. \tag{4.5}$$

With (4.4) and (4.5), we have

$$E(L) = \frac{\mu}{\mu - \lambda(m-1)}, \tag{4.6}$$

and

$$\rho_d = \frac{\lambda}{\mu - \lambda(m-1)}. \tag{4.7}$$

Substituting m in (4.3) with $E(L)$, we get

$$E(T_{dps}) = \frac{1}{\mu - \lambda(m-1)} + \frac{\frac{\lambda}{(\mu-\lambda(m-1))^2} + \frac{\rho}{\eta^2} + \frac{\gamma_m}{\kappa^2}}{\left(1 - \frac{\lambda}{\mu-\lambda(m-1)} - \frac{\rho}{\eta} - \frac{\gamma_m}{\kappa}\right)\left(1 - \frac{\rho}{\eta} - \frac{\gamma_m}{\kappa}\right)}. \tag{4.8}$$

We know that the static and the dynamic processor-sharing models are the same when $m = 1$ as $E(L) = 1$. As a consequence, (4.8) equals (4.3) when $m = 1$. From (4.6), $1 \leq E(L) \leq m$, which means that the dynamic processor-sharing model should improve performance over the static processor-sharing model.

4.2.4 Cyclic Polling Dispatch Model

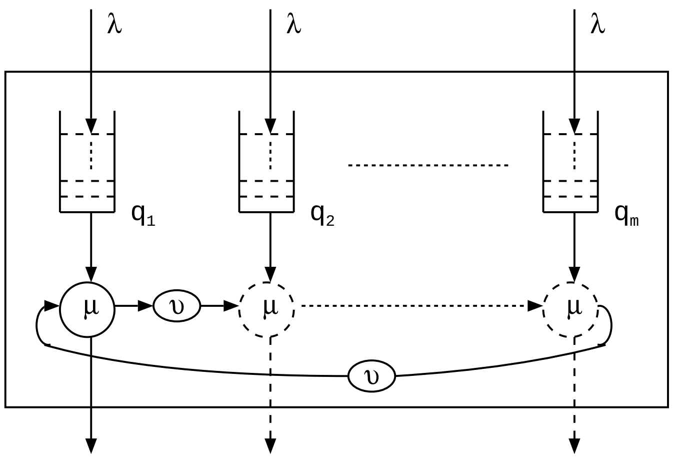


Figure 4.6: Cyclic polling dispatch model.

For the dynamic processor-sharing model, the dispatch facility is shared among queues which contain deliverable messages. If we dedicate the facility exclusively for one queue at a time and make the facility serve each queue in a sequential and cyclic turn, the cyclic polling dispatch model comes into view [124]. When the dispatch facility visits a message queue, if the queue contains a deliverable message, it will pick up the message from the head of the queue and dispatch it to its MH, and then turn to the next message queue with a switchover time S, an exponentially distributed r.v. with parameter υ. If there are no deliverable messages in the queue, it will visit the next message queue also with a switchover time S. With a switchover time, we remove the infinite loop when no messages are deliverable in all queues. Figure 4.6 shows this dispatch model. Obviously, when there is only one MH, no performance improvement will be achieved with the cyclic polling model, as the handoff or recovery time cannot be hidden by dispatching message for other MHs. Thus it reduces to the basic dispatch model.

4.2.5 Feedback Dispatch Model

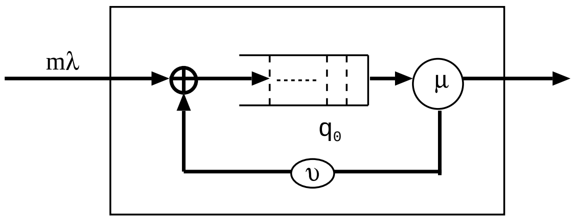


Figure 4.7: Feedback dispatch model.

Another way to reduce the blocked overhead forced by the message at the head of the queue in the basic model is to move the blocked message away and to yield the delivery chance to the subsequent messages. The removed message should be fed back to the queue, which introduces the feedback dispatch model, as shown in Figure 4.7. In this model, each removed message is added to the tail of the queue in its original turn. We also assume that a switchover time S is associated with the process of loading and unloading messages from the dispatch facility. Analysis of this model is more difficult due to the fact that the message input to the queue q_0 is no longer a Poisson input, since the message arrivals join the queue via the feedback path [107]. A side-effect of this model is that it modifies the sequence of messages dispatched to an MH.

4.3 Simulations and Discussions

In this section we compare different dispatch models in order to observe the relationships among them and to determine which one is the best under different conditions according to our defined performance metric: the expected message sojourn time. Our experiments are carried out by simulations developed with C-Sim [62]. Some parameter values are not changed throughout this section[1] , and we give them here: $\kappa = 10^{-1}$, $\eta = 1$, $\rho = 10^{-3}$, $\upsilon = 10^{-4}$. The handoff operation is a mobile characteristic provided by wireless networks and only contains some message exchanges, while the recovery procedure includes an additional state recovery. Therefore, the expected handoff completion time should be less than the expected recovery time.

Figure 4.8 shows the results of expected message sojourn time at AB when the number of MHs, m, increases. The simulation results are shown with solid lines, and analytical results are displayed with dash-dot lines. Different dispatch models demonstrate various relationships with the increase of the number of MHs. The sojourn times of the static processor-sharing model $E(T_{sps})$ and the cyclic polling model $E(T_{cp})$ exhibit moderate increases while the basic model $E(T_{ba})$ experiences sharp increase. The message sojourn times of the other two models, the dynamic processor-sharing model $E(T_{dps})$ and the feedback model $E(T_{fb})$, display almost no increase.

The dispatch capacity dedicated to a certain MH decreases with m in the

[1]Here we engage exponential distribution for simulation; however, it could be further extended to general distribution.

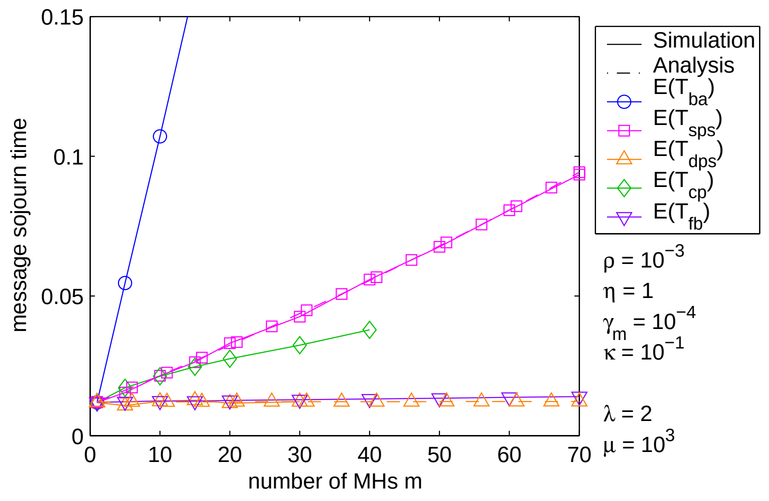


Figure 4.8: Expected message sojourn time vs. number of MHs.

static processor-sharing models, which induces the increase of the sojourn time; however, dynamic scheduling compensates part of loss of the dispatch capacity. It confirms that the dynamic processor-sharing model can be utilized advantageously. The cyclic polling model and the feedback model are different with other models in that they give dispatch opportunities to other deliverable messages instead of blocking them by yielding the dispatch facility exclusively to a deliverable message. They hide the unavailable periods by dispatching messages for other MHs. However, the cyclic polling model serves each MH only one message in each cycle, thus, the round time to the next dispatch opportunity may exceed the blocking time, which cancels out the advantage and thus the expected sojourn time increases. One approach to improve this model is to allow the dispatch facility to continuously serve each queue until it empties or an undeliverable message appears at the head of the queue. The little variation of the message sojourn times of the dynamic processor-sharing model and the feedback model is due to the small queue traffic intensity, as even when $m = 10$, ρ_a only has 0.14. They should also increase with m. From this figure, we can see that when $m = 1$, all models take the same effect on the expected message dispatch time. Actually under this condition, all the other four models are reduced to the basic model.

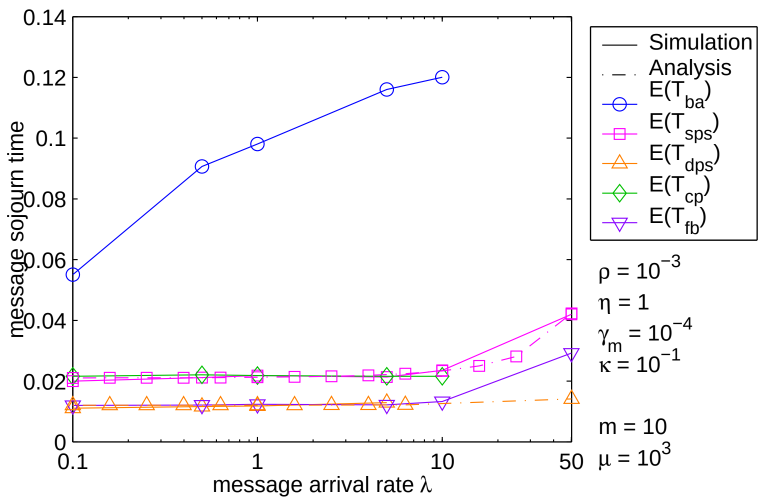


Figure 4.9: Expected message sojourn time vs. message arrival rate.

The message arrival rate λ produces an increase in message sojourn time for all the five models, as shown in Figure 4.9. This is obvious as the higher the message arrival rate is, the queue's traffic intensity increases, thus increasing the message waiting time.

We next show how the MH's failure rate affects the expected sojourn time in Figure 4.10. The sojourn times of all the five models also increase with the failure rate γ_m. The higher the MH's failure rate is, the greater the undeliverable probability for a message is. As the failure rate increases, the performance of the basic model diverges with those of other four models. Another phenomenon is that with a high MH's failure arrival rate, the analytical results show larger message sojourn times than the simulation results for the static and dynamic processor sharing models. This may due to the assumption in the head-of-the-line priority queue that MH's failures and handoffs are independent event sources with rate γ_m and ρ, respectively. However, in simulations

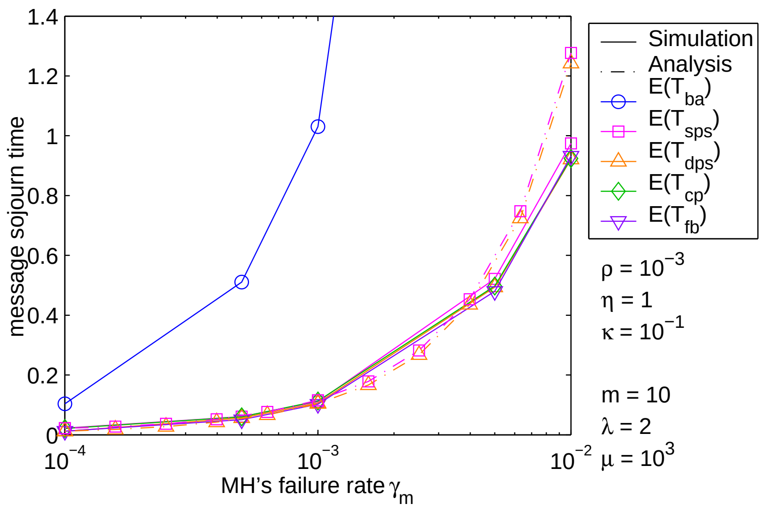


Figure 4.10: Expected message sojourn time vs. MH's failure rate.

we utilize the Markov model shown in Figure 4.1. Therefore, more failure or handoff events will be injected into the priority queue, resulting in a larger message sojourn time. With small γ_m or ρ, this difference is undiscernable. In the dynamic processor-sharing model, when the failure arrival rate is high enough, the dispatch facility is occupied by one message exclusively with a large probability. These observations can also be made with the handoff rate ρ. Prolonging the recovery period $E(U)$ or the handoff period $E(H)$ also apparently increases the message sojourn time.

With different MH's failure rates, some models exhibit differently relative behaviors with the expected message dispatch requirement $E(D) = 1/\mu$ which indicates the capacity of the dispatch facility, as shown in Figure 4.11. Increasing the dispatch requirement adds the sojourn time in all the five models. Comparing Figures 4.11(a) and (b), we observe that when the failure rate is low, increasing the dispatch requirement may even cause the static processor-sharing model performs worse than the basic model (larger message sojourn time). While with a high failure rate, it performs better than the basic model.

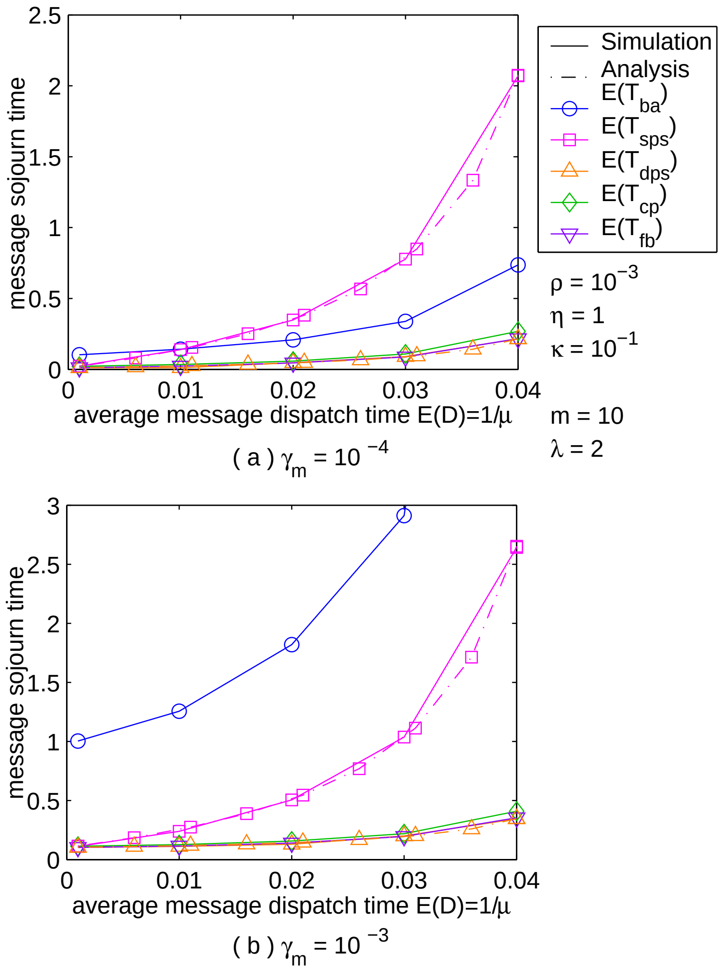


Figure 4.11: Expected message sojourn time vs. expected message dispatch requirement.

From all of the above figures, the basic model demonstrates the worst performance, and the static processor-sharing model exhibits the second worst performance. The other three models outperform these two models. We note, however, that there is only one queue in the basic and the feedback models, and the dispatch rate is reduced with m in the static processor-sharing model. When the number of MHs is large, the message arrival rate should be small; otherwise the queue in these three models will not be stable.

4.4 Summary

In this chapter, we perform analyses for message sojourn time at AB in the presence of MH failures and handoffs in wireless networks. To see how different message dispatch strategies influence the sojourn time, we study five dispatch models: the basic model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model. We derive the expected message sojourn time under steady state for the static and the dynamic processor-sharing models. Analytical and simulation results show that the basic model and the static processor-sharing dispatch model demonstrate the worst performance. The other three models may be suitable for applications as the dispatch strategy for AB; however, the runtime environment determines which one should be implemented.

□ **End of chapter.**

Chapter 5

Program Execution Time at Mobile Host

Program execution times with and without checkpointing in the presence of failures have been analyzed by many researchers. The derived execution time expressions are based on a given time requirement for a program executed on a stand-alone host without failures. In this chapter, we extend the analysis to mobile wireless environments. On account of the underlying message-passing communication mechanism, we employ the number of received computational messages instead of time to measure the progress of program execution on a mobile host. Handoff and wireless link failures are other distinct factors that should be taken into consideration. Three checkpointing strategies, deterministic, random, and time-based checkpointing, are exploited. These three checkpointing strategies have been employed in providing FT for wireless CORBA (Page 40). In our approach [28], failures may occur during checkpointing and recovery periods. We derive the LST of the c.d.f. of the program execution time and its expectation. For all three checkpointing strategies, we address the trade-off conditions against no checkpointing and the optimal checkpointing frequencies that maximize the average effectiveness. We also perform a number of assessments on the derived program execution times, and identify some critical parameters for future investigations.

5.1 Assumptions and Notations

We assume that the program at an MH determines its termination according to the number of computational messages that it should receive. Let n be the required number of messages. The AB disseminates computational messages to the MH with exponentially distributed time intervals. The expected inter-message arrival time is $1/\lambda$. During the execution of a program, four events may occur: MH failure, wireless link failure, handoff, and checkpointing, denoted as f, l, h, and c, respectively. The instances of the occurrences of MH and wireless link failures and handoffs form homogeneous Poisson processes with parameter γ_m, γ_l, and ρ, respectively.

Figure 5.1(a) shows the MH's state transition during program execution, which extends the state transition diagram in Figure 4.1. States with dashed circles indicate that wireless communications are engaged in these states. State 0 is the normal (operational) state, in which messages can be received with rate λ^* in the presence of wireless link failures. If a handoff occurs, the program transits to state 1. State 3 is the checkpointing state, and ι is the checkpointing rate, which will be specified in the following different checkpointing strategies. The program will enter state 2 if an MH failure occurs when the program in state 0, 1, or 3. States 3 and 2 are composite states, whose detailed representations are shown in Figure 5.1(b) and 5.1(c), respectively. Successfully creating a checkpoint needs two steps: taking a checkpoint first (state 4) and then saving the checkpoint on stable storage (state 5). The required time for these two substates are T_1 and T_2, respectively. We have stated in Chapter 3 that the MH is not suitable to be treated as stable storage, and that the checkpoint will be saved on its currently associated AB [24, 44, 94, 105, 138]. Therefore,

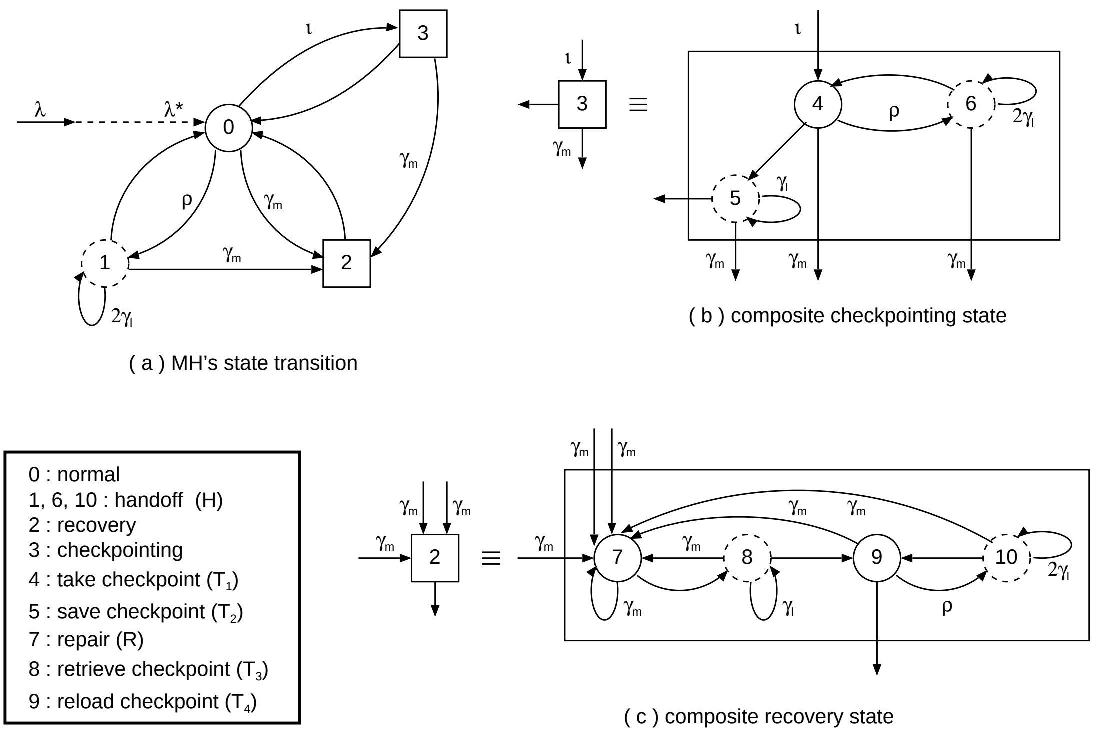

λ
λ*
ι
ρ
γm
2γl
(a) MH's state transition
(b) composite checkpointing state
(c) composite recovery state
0 : normal
1, 6, 10 : handoff (H)
2 : recovery
3 : checkpointing
4 : take checkpoint (T1)
5 : save checkpoint (T2)
7 : repair (R)
8 : retrieve checkpoint (T3)
9 : reload checkpoint (T4)

Figure 5.1: MH's state transition with checkpointing.

wireless communications are engaged in the checkpoint saving substate. As a result, a wireless link failure will cause the saving operation to be retried. State 2 means that the MH will undergo repair and rollback processes following an MH failure. The *repair* process brings the failed MH back to normal operation (state 7), and the *rollback* process retrieves the program status saved in the most recent checkpoint from a remote storage via wireless link (state 8) and reloads the status onto the MH, preparing it to receive subsequent messages (state 9). The required repair time is denoted as R, the required time to retrieve a checkpoint as T_3, and the required time to reload a checkpoint as T_4. Let H be the handoff time. All these six time requirements, R, H, and T_i, $i = 1, 2, 3, 4$, are regarded as independent and random variables with general distributions. During handoff, the MH should maintain two wireless links: one with the OAB and the other with the NAB; therefore, the link failure rate on handoff states (states 1, 6, and 10) is $2\gamma_l$. For the pessimistic strategy (Page 19), H includes the time of transferring the most recent checkpoint; otherwise, this time will be considered as a part of T_3.

The MH cannot perform handoffs during the repair state since in this state the MH is not ready to exchange any information with ABs. Since states 5 and 8 transfer checkpoints via wireless links, these two states are not allowed to be interrupted by handoffs. If the MH is not in the operational state, i.e., the MH is in the handoff, checkpointing, or recovery states, no computational messages will be forwarded to the MH. In the repair state, we take into consideration the possibility that the MH might be disconnected from the network for a while when it fails. Failures will occur irrespective of the MH's state, and will be assumed immediately detected upon their occurrences, which implies that the state saved by a checkpoint is always correct, and that a program always terminates with correct results [76]. After the MH's rollback period, its associated AB automatically redistributes logged messages since the most recent checkpoint. The logged messages may be dispersed in different ABs with the lazy and trickle strategies (Page 19), and so they need time to be collected and redispatched; therefore, we assume that after a failure the inter-message arrival events still follow the same process as before.

Let $X(n)$ be the total program execution time with n messages in the absence of MH and wireless link failures, handoffs, and checkpointings. We define that $Z^{(e)}$ represents the total program execution time in the presence of event e, $e = f, l, h, c$, with the time requirement Z. Multiple event types may be presented during the program execution. For example, $Z^{(f,h)}$ denotes the program execution time if MH failures and handoffs both occur during the execution. The general cumulative distribution function (c.d.f.) of a r.v. Z is $G_Z(t)$. $\phi_Z(s)$ denotes the Laplace-Stieltjes Transform (LST) of $G_Z(t)$, so $\phi_Z(s) = \int_{t=0}^{\infty} e^{-st} dG_Z(t) = E(e^{-sZ})$. Let $\tilde{Z}$ be a r.v. which has the same probability distribution as Z, then $\phi_{\tilde{Z}}(s) = \phi_Z(s)$. Based on the above assumptions, we first derive some preliminary execution times, and then describe three checkpointing strategies in the following sections.

5.2 Preliminary Execution Times

Without failures, the required execution times for performing handoff, checkpointing, and recovery are H, $T_1 + T_2$, and $T_3 + T_4$, respectively. With failures, the induced execution times are prolonged. In this section, we will derive these execution times in the presence of MH or wireless link failures.

5.2.1 Handoff Time with Wireless Link Failures

The handoff time requirement is H. If a wireless link failure occurs, the MH should retry its communication with the corresponding AB – a further attempt may be successful, because of the intermittent characteristic of wireless link failures. Thus, the resulting handoff time is $H^{(l)}$.

Lemma 1 The LST of the c.d.f. of $H^{(l)}$, the handoff time in the presence of wireless link failures with the time requirement H, i.e., the sojourn time in state 1, 6, or 10, is given by

$$\phi_{H^{(l)}}(s) = \frac{(s + 2\gamma_l)\phi_H(s + 2\gamma_l)}{s + 2\gamma_l \phi_H(s + 2\gamma_l)}, \tag{5.1}$$

and the expectation of $H^{(l)}$ is

$$E\left[H^{(l)}\right] = \frac{1 - \phi_H(2\gamma_l)}{2\gamma_l \phi_H(2\gamma_l)}. \tag{5.2}$$

Proof: Let Y be the time to the first wireless link failure after starting handoff, then we have

$$H^{(l)} = \begin{cases} H & : \ if \quad H < Y \\ Y + \tilde{H}^{(l)} & : \ otherwise \end{cases}.$$

If $H < Y$, the handoff will be successfully completed without link failures, so the handoff time is H. Otherwise when $H \geq Y$, a link failure occurs after which the MH retries its handoff operation, denoted as $\tilde{H}^{(l)}$. Thus the total handoff time is $Y + \tilde{H}^{(l)}$ in this case. Taking the conditional expectation of $H^{(l)}$, we get

$$E\left[e^{-sH^{(l)}}|H, Y\right] = \begin{cases} e^{-sH} & : \ if \quad H < Y \\ e^{-sY} E\left[e^{-s\tilde{H}^{(l)}}\right] & : \ otherwise \end{cases},$$

as Y should be independent of $\tilde{H}^{(l)}$. Removing the condition on Y, we have

$$\begin{aligned} E\left[e^{-sH^{(l)}}|H\right] &= \int_{y=0}^{\infty} E\left[e^{-sH^{(l)}}|H, Y = y\right] \cdot 2\gamma_l e^{-2\gamma_l y} dy \\ &= e^{-(s+2\gamma_l)H} + \frac{2\gamma_l E\left[e^{-s\tilde{H}^{(l)}}\right]}{s + 2\gamma_l} \left[1 - e^{-(s+2\gamma_l)H}\right]. \end{aligned}$$

Removing the condition on H, the result is

$$\phi_{H^{(l)}}(s) = \phi_H(s + 2\gamma_l) + \frac{2\gamma_l \phi_{H^{(l)}}(s)}{s + 2\gamma_l} \left[1 - \phi_H(s + 2\gamma_l)\right]. \tag{5.3}$$

Rearranging the above equation yields (5.1). After engaging the moment generating property of the Laplace transform, $E(Z) = -d\phi_Z(s)/ds|_{s=0}$ [112], the expected handoff time in the presence of wireless link failures is given by (5.1).

■

With no wireless link failures during handoff, i.e., when γ_l tends to 0, the expectation of the handoff time is reduced to

$$\lim_{\gamma_l \to 0} E\left[H^{(l)}\right] = E(H) \tag{5.4}$$

by applying L'Hopital's rule on (5.1), which confirms the correctness of the derivation of (5.1).

5.2.2 Checkpointing Time with Handoffs and Wireless Link Failures

During the period of taking a checkpoint, the MH may undergo handoffs; however, no handoffs are allowed to be performed in the state of saving a checkpoint. Therefore, the total checkpointing time, i.e., the sojourn time in state 3, is $T_1^{(h,l)} + T_2^{(l)}$, denoted as C for simplicity.

Lemma 2 The LST of the c.d.f. of C, the sojourn time in state 3, is given by

$$\phi_C(s) = \phi_{T_1^{(h,l)}}(s) \cdot \phi_{T_2^{(l)}}(s), \tag{5.5}$$

in which

$$\phi_{T_1^{(h,l)}}(s) = \phi_{T_1}(s + \rho - \rho\phi_{H^{(l)}}(s)), \tag{5.6}$$

and

$$\phi_{T_2^{(l)}}(s) = \frac{(s+\gamma_l)\phi_{T_2}(s+\gamma_l)}{s+\gamma_l\phi_{T_2}(s+\gamma_l)}. \tag{5.7}$$

Proof: $T_1^{(h,l)}$ can be expressed by $T_1^{(h,l)} = T_1 + \sum_{j=1}^{K} H_j^{(l)}$, in which K denotes the number of handoffs in the presence of link failures during a checkpointing period. With $Pr(K = k|T_1) = (\rho T_1)^k e^{-\rho T_1}/k!$, it is easy to derive the result of $\phi_{T_1^{(h,l)}}(s)$, as shown in (5.6). The derivation process of $T_2^{(l)}$ is similar as that of (5.1); however, the link failure rate is now γ_l, instead of $2\gamma_l$, as only one wireless link is engaged in saving a checkpoint. (5.5) can be directly derived from the independence between $T_1^{(h,l)}$ and $T_2^{(l)}$. ■

5.2.3 Recovery Time with Handoffs and MH and Wireless Link Failures

As checkpoints are saved remotely on ABs, the checkpoint retrieval time through wireless links after a repair process is also considerable. So the total time between a failure and the instant that the program is ready to receive computational messages is $[R + T_3^{(l)} + T_4^{(h,l)}]^{(f)}$, denoted as R' for simplicity.

Lemma 3 The LST of the c.d.f. of R', the recovery time in the presence of handoffs and MH and wireless link failures, i.e., the sojourn time in state 2, is given by

$$\phi_{R'}(s) = \frac{(s+\gamma_m)\phi_R(s+\gamma_m)\phi_{T_3^{(l)}}(s+\gamma_m)\phi_{T_4^{(h,l)}}(s+\gamma_m)}{s+\gamma_m\phi_R(s+\gamma_m)\phi_{T_3^{(l)}}(s+\gamma_m)\phi_{T_4^{(h,l)}}(s+\gamma_m)}, \tag{5.8}$$

and the expectation of R' is

$$E(R') = \frac{1-\phi_R(\gamma_m)\phi_{T_3^{(l)}}(\gamma_m)\phi_{T_4^{(h,l)}}(\gamma_m)}{\gamma_m\phi_R(\gamma_m)\phi_{T_3^{(l)}}(\gamma_m)\phi_{T_4^{(h,l)}}(\gamma_m)}. \tag{5.9}$$

Proof: The recovery time requirement is $R + T_3^{(l)} + T_4^{(h,l)}$. If a failure occurs on the MH, the MH will enter the repair state irrespective of the MH's previous state. This is a retry process, similar to the handoff operation in the presence of link failures; therefore, following the similar derivation steps in Lemma 1, we can obtain (5.8) and (5.9). Here $\phi_{T_3^{(l)}}(s)$ and $\phi_{T_4^{(h,l)}}(s)$ have similar forms to those of (5.7) and (5.6), however, with different time requirements, T_3 and T_4, respectively. ■

5.3 Deterministic Checkpointing Strategy

The first two checkpointing strategies in this chapter, deterministic and random checkpointing, are message-number-based checkpointing strategies, which place checkpoints according to the number of received computational messages. The difference between these two strategies is how the number of messages in each checkpointing interval is distributed. By deterministic we mean that the number of messages in a checkpointing interval is fixed, and we denote this number as u. If the message arrival rate is λ, the checkpointing rate $\iota_{dc} = \lambda/u$. With u chosen in advance, the program execution is broken into $w = \max(\left\lceil \frac{n}{u} \right\rceil, 1)$ intervals. Each of the first $(w-1)$ intervals receives u messages and takes a checkpoint, and the last interval should receive the residual messages; however, no checkpoints will be taken in the last interval. Consequently, there are a total of $(w-1)$ checkpoints. The number of messages in the last interval is denoted as $u' = n - u \cdot (\left\lceil \frac{n}{u} \right\rceil - 1)$, $0 < u' \leq u$.

Without wireless link failures, the AB disseminates computational messages to the MH with exponentially distributed time intervals at rate λ. Note that the message arrival rate is still λ in the presence of link failures, due to the memoryless property of the exponential distribution. Therefore, $\lambda^* = \lambda$.

We now present a theorem for the program execution time with deterministic checkpointing.

Theorem 1 Let

$$Q_1(s) = \frac{\lambda}{s + \gamma_m + \lambda + \rho - \rho\phi_{H^{(l)}}(s+\gamma_m)}, \tag{5.10}$$

and

$$q_1 = Q_1(s)|_{s=0} = \frac{\lambda}{\gamma_m + \lambda + \rho - \rho\phi_{H^{(l)}}(\gamma_m)}, \tag{5.11}$$

then the LST of the c.d.f. of $X^{(dc,f,h,l)}(n,u)$, the program execution time with deterministic checkpointing, with the checkpointing parameter u, contains the form

$$\begin{aligned} \phi_{X^{(dc,f,h,l)}}(s,n,u) &= \left[\frac{(s+\gamma_m)\phi_C(s+\gamma_m)Q_1^u(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)(1-\phi_C(s+\gamma_m)Q_1^u(s))}\right]^{w-1} \\ &\quad \cdot \left[\frac{(s+\gamma_m)Q_1^{u'}(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)(1-Q_1^{u'}(s))}\right], \end{aligned} \tag{5.12}$$

and its expectation is given by

$$E\left[X^{(dc,f,h,l)}(n,u)\right] = \left[\frac{1}{\gamma_m} + E(R')\right]\left[(w-1)\left(\frac{q_1^{-u}}{\phi_C(\gamma_m)} - 1\right) + (q_1^{-u'} - 1)\right]. \tag{5.13}$$

Proof: With deterministic checkpointing, the execution times for the i^{th} interval, $i = 1, 2, \ldots, (w-1)$, are independent and identically distributed (i.i.d.) r.v., and each of these intervals contains a checkpoint creation period. Therefore, we can analyze the execution times for these intervals first. Let K_i denote the number of handoffs during the normal execution time period and Y_i denote the time to the first MH failure in the i^{th} interval. Then we get

$$X_i^{(dc,f,h,l)}(u) = \begin{cases} X_i(u) + \sum_{j=1}^{K_i} H_j^{(l)} + C & : \; if \;\; Y_i > X_i(u) + \sum_{j=1}^{K_i} H_j^{(l)} + C \\ Y_i + R' + \tilde{X}_i^{(dc,f,h,l)}(u) & : \; otherwise \end{cases}.$$

If $Y_i > X_i(u) + \sum_{j=1}^{K_i} H_j^{(l)} + C$, then the program will successfully complete the i^{th} interval by creating a checkpoint without undergoing any MH failures. If $Y_i \le X_i(u) + \sum_{j=1}^{K_i} H_j^{(l)} + C$, the MH fails before the program successfully creates a checkpoint. Then the MH will undergo repairs and rollbacks and the program should be restarted from the state saved on the most recent checkpoint, after which the program should receive u computational messages without MH failure interruption again. The time this takes is another r.v. $\tilde{X}_i^{(dc,f,h,l)}(u)$; however, it engages the same probability distribution as $X_i^{(dc,f,h,l)}(u)$. Taking the conditional expectation of $X_i^{(dc,f,h,l)}(u)$, we get

$$E\left[e^{-sX_i^{(dc,f,h,l)}(u)} | X_i(u), K_i, H_j^{(l)}, C, Y_i\right]$$
$$= \begin{cases} e^{-sX_i(u)} \cdot \prod_{j=1}^{K_i} e^{-sH_j^{(l)}} \cdot e^{-sC} & : \; if \;\; Y_i > X_i(u) + \sum_{j=1}^{K_i} H_j^{(l)} + C \\ e^{-sY_i} E(e^{-sR'}) E[e^{-s\tilde{X}_i^{(dc,f,h,l)}(u)}] & : \; otherwise \end{cases},$$

in which $H_j^{(l)}, 1 \leq j \leq K_i$ are i.i.d. r.v.s and $X_i(u)$ inherits an u-stage Erlang distribution with parameter λ. Removing the conditions backwards one by one and solving the resulting equation, we get

$$\phi_{X_i^{(dc,f,h,l)}}(s,u) = \frac{(s+\gamma_m)\phi_C(s+\gamma_m)Q_1^u(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)\left[1-\phi_C(s+\gamma_m)Q_1^u(s)\right]}. \tag{5.14}$$

After engaging the moment generating property of the Laplace transform, its expectation is given by

$$E\left[X_i^{(dc,f,h,l)}(u)\right] = \left[\frac{1}{\gamma_m} + E(R')\right]\left[\frac{q_1^{-u}}{\phi_C(\gamma_m)} - 1\right].$$

The last interval is different from other intervals as no checkpoint will be established and the required number of messages is u', instead of u; therefore, its execution time can be expressed as

$$X_w^{(dc,f,h,l)}(u') = \begin{cases} X_w(u') + \sum_{j=1}^{K_w} H_j^{(l)} & : \ if \ \ Y_w > X_w(u') + \sum_{j=1}^{K_w} H_j^{(l)} \\ Y_w + R' + \tilde{X}_w^{(dc,f,h,l)}(u') & : \ otherwise \end{cases}.$$

This is actually the execution time without checkpointing with message requirement u'. Following the same steps, we get the LST

$$\phi_{X_w^{(dc,f,h,l)}}(s,u') = \frac{(s+\gamma_m)Q_1^{u'}(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)\left[1-Q_1^{u'}(s)\right]}, \tag{5.15}$$

and its expectation

$$E\left[X_w^{(dc,f,h,l)}(u')\right] = \left[\frac{1}{\gamma_m} + E(R')\right]\left[q_1^{-u'} - 1\right].$$

As the total execution time is the summation of the execution times of w separate parts,

$$X^{(dc,f,h,l)}(n,u) = \sum_{i=1}^{w-1} X_i^{(dc,f,h,l)}(u) + X_w^{(dc,f,h,l)}(u'),$$

then we have

$$\phi_{X^{(dc,f,h,l)}}(s,n,u) = \left[\phi_{X_i^{(dc,f,h,l)}}(s,u)\right]^{w-1} \cdot \phi_{X_w^{(dc,f,h,l)}}(s,u'),$$

and

$$E\left[X^{(dc,f,h,l)}(n,u)\right] = (w-1)E\left[X_i^{(dc,f,h,l)}(u)\right] + E\left[X_w^{(dc,f,h,l)}(u')\right].$$

Finally, we obtain (5.12) and (5.13). ■

The most evident difference between (5.14) and (5.15) is that the former equation contains the term $\phi_C(s+\gamma_m)$ introduced by the checkpointing period. If we remove this checkpointing period by substituting $\phi_C(s+\gamma_m)$ with 1, (5.14) will be reduced to (5.15). Thus, the program execution time without checkpointing can be easily derived from Theorem 1, which is given by the following corollary.

Corollary 1 The LST of the c.d.f. of the program execution time without checkpointing, $X^{(f,h,l)}(n)$, contains the form

$$\phi_{X^{(f,h,l)}}(s,n) = \frac{(s+\gamma_m)Q_1^n(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)[1-Q_1^n(s)]}, \tag{5.16}$$

and the expectation of $X^{(f,h)}(n)$ is

$$E\left[X^{(f,h,l)}(n)\right] = \left[\frac{1}{\gamma_m}+E(R')\right](q_1^{-n}-1). \tag{5.17}$$

Proof: If $u \geq n$, the program will be terminated with no checkpoints established during execution. With $u \geq n$, $w=1$ and $u'=n$. Substituting these conditions into (5.12) and (5.13), (5.16) and (5.17) will be derived. ■

From (5.17), we know that the expectation of the program execution time varies exponentially with respect to the required message number n; however, (5.13) shows that, after engaging the deterministic checkpointing strategy, the execution time demonstrates a linear relationship with the number of checkpointing intervals m. Note that it still exhibits an exponential relationship with parameter u, the number of messages in a checkpointing interval. This confirms the results shown in Figure 3.5.

The derivations of Theorem 1 and Corollary 1 implicitly assume that the initial state of the program is saved remotely on an SH or AB, by which the time R' taken to recover the initial state is included in $X_1^{(dc,f,h,l)}(u)$ and $X^{(f,h,l)}(n)$. Therefore, this assumption simplifies our derivations.

With no failures during program execution, i.e., both γ_m and γ_l tending to 0, then the expectation of the execution time with deterministic checkpointing is

$$\lim_{\gamma_m,\gamma_l \to 0} E\left[X^{(dc,f,h,l)}(n,u)\right] = [1+\rho E(H)]\left[\frac{n}{\lambda} + (w-1)(E(T_1)+E(T_2))\right], \tag{5.18}$$

and the expectation of the execution time without checkpointing is

$$\lim_{\gamma_m,\gamma_l \to 0} E\left[X^{(f,h,l)}(n)\right] = [1+\rho E(H)]\frac{n}{\lambda}. \tag{5.19}$$

The deterministic checkpointing strategy can be extended to a more general strategy. Starting with a requirement n, the program continues to receive messages until the message number reaches u_1, at which point the strategy dictates a checkpoint [34]. The next checkpoint is triggered when a further u_2 messages have been received since the preceding checkpoint. Continuing in a similar way, this extended strategy specifies $(w-1)$ checkpoints and $u_1, u_2, \ldots, u_{(w-1)}, u_w$ messages in each interval, in which $u_w = n - u_1 - u_2 - \ldots - u_{(w-1)}$. The LST of the c.d.f. of the program's total execution time is then given by

$$\begin{aligned}\phi_{X^{(edc,f,h,l)}}(s,n,u) &= \frac{(s+\gamma_m)Q_1^{u_w}(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)[1-Q_1^{u_w}(s)]} \\ &\cdot \prod_{i=1}^{w-1} \frac{(s+\gamma_m)\phi_C(s+\gamma_m)Q_1^{u_i}(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)[1-\phi_C(s+\gamma_m)Q_1^{u_i}(s)]},\end{aligned} \tag{5.20}$$

and its expectation is given by

$$E\left[X^{(edc,f,h,l)}(n,u)\right] = \left[\frac{1}{\gamma_m} + E(R')\right]\left[\sum_{i=1}^{w-1}\left(\frac{q_1^{-u_i}}{\phi_C(\gamma_m)} - 1\right) + (q_1^{-u_w} - 1)\right]. \tag{5.21}$$

5.4 Random Checkpointing Strategy

The deterministic checkpointing strategy creates checkpoints according to a fixed and predefined number of computational messages. It may be too rigid for adaptation to different conditions. Adjusting the message number in each checkpointing interval from a constant to a r.v. may afford some performance advantages. In order to explore this extension, we engage the strategy of random checkpointing. The random checkpointing strategy creates checkpoints when the program has received I messages since its preceding checkpoint, in which I is a r.v.. If I is generally distributed, it is difficult to get an analytical solution. Here we assume that I is a r.v. with a geometric distribution whose parameter is p, and whose probability mass function (p.m.f.) is $Pr(I = i) = p(1-p)^{i-1}$, in which $i = 1, 2, \ldots$. Thus the checkpointing rate $\iota_{rc} = \lambda p$.

Now we derive a theorem for the program execution time with random checkpointing as follows:

Theorem 2 Let

$$Q_2(s) = \frac{p\phi_C(s+\gamma_m)}{1-(1-p)Q_1(s)}, \tag{5.22}$$

and

$$q_2 = Q_2(s)|_{s=0} = \frac{p\phi_C(\gamma_m)}{1-(1-p)q_1}, \tag{5.23}$$

then the LST of the c.d.f. of $X^{(rc,f,h,l)}(n,p)$, the program execution time with random checkpointing, with the checkpointing parameter p, can be expressed in the form

$$\begin{aligned}
&\phi_{X^{(rc,f,h,l)}}(s,n,p)\\
=\; &\frac{(s+\gamma_m)Q_1(s)}{s+\gamma_m-\gamma_m\phi_{R'}(s)[1-Q_1(s)]}\\
&\cdot\prod_{i=2}^{n}\left\{1+\frac{[s+\gamma_m-\gamma_m\phi_{R'}(s)][(p\phi_C(s+\gamma_m)-p+1)Q_1(s)-1]}{s+\gamma_m-\gamma_m\phi_{R'}(s)[1-Q_2(s)Q_1(s)+(Q_2(s)-1)(1-p)^{i-1}Q_1^i(s)]}\right\},
\end{aligned} \tag{5.24}$$

and its expectation is given by

$$E\left[X^{(rc,f,h,l)}(n,p)\right]=\left[\frac{1}{\gamma_m}+E(R')\right]\left[q_1^{-1}-1+\sum_{i=2}^{n}\frac{q_1^{-1}-p\phi_c(\gamma_m)+p-1}{q_2+(1-q_2)((1-p)q_1)^{i-1}}\right]. \tag{5.25}$$

Proof: We utilize the difference equation to solve this problem. Let K be the number of handoffs during the normal execution without checkpointing, K' be the number of handoffs before taking the first checkpoint, and Y be the time to the first MH failure. Then the conditional execution time can be expressed by

$$X^{(rc,f,h,l)}(n,p)=\begin{cases}
X(n)+\sum_{j=1}^{K}H_j^{(l)} \\
\qquad : \ if \ \ Y>X(n)+\sum_{j=1}^{K}H_j^{(l)} \ \ and \ \ I\geq n\\
Y+R'+\tilde{X}^{(rc,f,h,l)}(n,p)\\
\qquad : \ if \ \ (Y\leq X(n)+\sum_{j=1}^{K}H_j^{(l)} \ \ and \ \ I\geq n)\\
\qquad \ \ or \ \ (Y\leq X(I)+\sum_{j=1}^{K'}H_j^{(l)}+C \ \ and \ \ I<n)\\
X(I)+\sum_{j=1}^{K'}H_j^{(l)}+C+X^{(rc,f,h,l)}(n-I,p)\\
\qquad : \ if \ \ Y>X(I)+\sum_{j=1}^{K'}H_j^{(l)}+C \ \ and \ \ I<n
\end{cases}.$$

If $(Y > X(n)+\sum_{j=1}^{K}H_j^{(l)}$ *and* $I\geq n)$, before taking any checkpoints the program will terminate successfully without MH failures, so the total execution time is $X(n)+\sum_{j=1}^{K}H_j^{(l)}$. If $(Y\leq X(n)+\sum_{j=1}^{K}H_j^{(l)}$ *and* $I\geq n)$, an MH failure occurs before the program receives n messages and until this failure time instant no checkpoints have been established; while if $(Y \leq X(I) + \sum_{j=1}^{K'} H_j^{(l)} + C \; and \; I < n)$, an MH failure occurs before the program successfully creates its first checkpoint. Under both conditions, the MH needs repair and rollback, after which the program should receive n messages all over again. If $(Y > X(I) + \sum_{j=1}^{K'} H_j^{(l)} + C \; and \; I < n)$, a checkpoint will be created after receiving I messages and no MH failures have occurred during this stage, after which $(n - I)$ messages still need to be received. So the total time is $X(I) + \sum_{j=1}^{K'} H_j^{(l)} + C + X^{(rc,f,h,l)}(n - I, p)$. Taking LST and removing the conditions on $Y, C, H_j^{(l)}, K, K', X(n)$, and $X(I)$ one by one, we get

$$\begin{aligned}
&\phi_{X^{(rc,f,h,l)}}(s, n, p) \\
= \; &(1-p)^{n-1} Q_1^n(s) + \left[B_1(s,p) - (1-p)^{n-1} Q_1^n(s) B_2(s,p)\right] \phi_{X^{(rc,f,h,l)}}(s, n, p) \\
&+ p\phi_C(s + \gamma_m) \sum_{i=1}^{n-1} (1-p)^{i-1} Q_1^i(s) \phi_{X^{(rc,f,h,l)}}(s, n-i, p), \qquad (5.26)
\end{aligned}$$

in which

$$B_1(s,p) = \frac{\gamma_m \phi_{R'}(s)}{s + \gamma_m} \left[1 - \frac{p\phi_C(s+\gamma_m) Q_1(s)}{1 - (1-p)Q_1(s)}\right],$$

and

$$B_2(s,p) = \frac{\gamma_m \phi_{R'}(s)}{s + \gamma_m} \left[1 - \frac{p\phi_C(s+\gamma_m)}{1 - (1-p)Q_1(s)}\right].$$

Taking the z-transform with respect to n,

$$\begin{aligned}
&\phi^*_{X^{(rc,f,h,l)}}(s, z, p) \\
= \; &\sum_{n=0}^{\infty} \frac{\phi_{X^{(rc,f,h,l)}}(s, n, p)}{z^n} \\
= \; &\frac{z}{(1-p)(z - (1-p)Q_1(s))} + B_1(s,p) \phi^*_{X^{(rc,f,h,l)}}(s, z, p) \\
&- \frac{B_2(s,p)}{1-p} \cdot \phi^*_{X^{(rc,f,h,l)}}\left(s, \frac{z}{(1-p)Q_1(s)}, p\right) \\
&+ \frac{p\phi_C(s+\gamma_m)}{1-p} \left[\frac{z}{(1-p)(z - (1-p)Q_1(s))} - 1\right] \phi^*_{X^{(rc,f,h,l)}}(s, z, p).
\end{aligned}$$

After some manipulations and taking the reverse z-transform, we obtain

$$\begin{aligned} & \left[(1-B_1(s,p))(1-p)+B_2(s,p)(1-p)^n Q_1(s)^n\right]\phi_{X^{(rc,f,h,l)}}(s,n,p) \\ = \; & \delta_0(n)+\phi_{X^{(rc,f,h,l)}}(s,n-1,p) \\ & \cdot\left[((1-B_1(s,p))(1-p)+p\phi_C(s+\gamma_m))(1-p)Q_1(s)+B_2(s,p)(1-p)^n Q_1(s)^n\right], \end{aligned} \tag{5.27}$$

in which $\delta_0(n)$ is the Dirac delta function. As the message requirement n is always greater than or equal to 1, $\delta_0(n)=0$. Substituting $n=1$ to (5.26), we get

$$\phi_{X^{(rc,f,h,l)}}(s,1,p)=\frac{(1-p)Q_1(s)}{(1-B_1(s,p))(1-p)+B_2(s,p)(1-p)Q_1(s)}.$$

Thus, (5.27) can be solved, and with some simplifications, we get (5.24). The corresponding expectation can also be derived using the moment generating property. ■

Note that (5.25) shows that the execution time is linearly related to the message requirement n. We also note that with no failures during program execution, i.e., both γ_m and γ_l tending to 0, then the expectation of the execution time with random checkpointing is

$$\lim_{\gamma_m,\gamma_l\to 0} E\left[X^{(rc,f,h,l)}(n,p)\right]=[1+\rho E(H)]\left[\frac{n}{\lambda}+p(n-1)(E(T_1)+E(T_2))\right]. \tag{5.28}$$

From (5.28), it is easily observed that the expected checkpoint number with the random checkpointing strategy is given by $E[N_{rc}(n,p)]=p(n-1)$, whereas with the deterministic checkpointing strategy, the checkpoint number is $N_{dc}(n,u)=(w-1)$. When $u=p^{-1}$, $E[N_{rc}(n,p)]\geq N_{dc}(n,u)$, which indicates that on average the random checkpointing takes more checkpoints than the deterministic checkpointing with the same checkpointing rate.

It is noted that when $p=0$, which means that the random checkpointing strategy is reduced to the strategy without checkpointing, (5.24) and (5.25) will contain the same forms as (5.16) and (5.17), respectively. When $n = 1$, this random checkpointing strategy is also reduced to the strategy without checkpointing. This occurs because, when the program receives the one message, it terminates successfully; thus no checkpoints will be created, and the probability of checkpointing before receiving any messages is 0 by definition.

5.5 Time-based Checkpointing Strategy

The above two checkpointing strategies, deterministic and random checkpointing, place checkpoints with respect to the message arrival events. We know that the checkpoint placement strategies based on the number of messages received may lead to a bad distribution of checkpoints with respect to time for applications in which program states are changed due not only to message exchanges but also to human interactions. Therefore, time-based strategies should also be considered and analyzed under the condition of terminating after receiving a certain number of messages. Let the checkpointing interval be a constant time v. Then the time-based checkpointing strategy will establish a checkpoint when the accumulated time in the computational state since the preceding checkpoint reaches v, where the checkpointing rate $\iota_{tc} = 1/v$.

The following theorem gives the program execution time with time-based checkpointing.

Theorem 3 Let

$$Q_3(s) = s + \gamma_m + \lambda + \rho - \rho\phi_{H^{(l)}}(s + \gamma_m), \tag{5.29}$$

$$q_3 = Q_3(s)|_{s=0} = \gamma_m + \lambda + \rho - \rho\phi_{H^{(l)}}(\gamma_m), \tag{5.30}$$

and

$$G(s, n, v) = Q_1^n(s)\left[1 - \phi_v(Q_3(s))\sum_{i=0}^{n-1}\frac{(vQ_3(s))^i}{i!}\right], \tag{5.31}$$

then the LST of the c.d.f. of $X^{(tc,f,h,l)}(n,v)$, the program execution time with time-based checkpointing, with the checkpointing parameter v, can be expressed as

$$\phi_{X^{(tc,f,h,l)}}(s,n,v) = \frac{G_1(s,n,v)}{G_2(s,n,v)}, \tag{5.32}$$

in which

$$\begin{aligned} & G_1(s,n,v) \\ = \; & (s+\gamma_m)\left[G(s,n,v) + \phi_C(s+\gamma_m)\phi_v(Q_3(s))\sum_{i=1}^{n-1}\frac{(\lambda v)^i}{i!}\phi_{X^{(tc,f,h,l)}}(s,n-i,v)\right] \end{aligned} \tag{5.33}$$

and

$$\begin{aligned} & G_2(s,n,v) \\ = \; & (s+\gamma_m)[1-\phi_C(s+\gamma_m)\phi_v(Q_3(s))] \\ & -\gamma_m\phi_{R'}(s)\left[1-G(s,n,v)-\phi_C(s+\gamma_m)\phi_v(Q_3(s))\sum_{i=0}^{n-1}\frac{(\lambda v)^i}{i!}\right], \end{aligned} \tag{5.34}$$

and its expectation is given by

$$\begin{aligned} E\left[X^{(tc,f,h,l)}(n,v)\right] = \; & \left[\frac{1}{\gamma_m}+E(R')\right]\left[\frac{1-\phi_C(\gamma_m)\phi_v(q_3)}{G(0,n,v)+\phi_C(\gamma_m)\phi_v(q_3)\sum_{i=1}^{n-1}\frac{(\lambda v)^i}{i!}}-1\right] \\ & + \frac{\phi_C(\gamma_m)\phi_v(q_3)\sum_{i=1}^{n-1}\frac{(\lambda v)^i}{i!}E\left[X^{(tc,f,h,l)}(n-i,v)\right]}{G(0,n,v)+\phi_C(\gamma_m)\phi_v(q_3)\sum_{i=1}^{n-1}\frac{(\lambda v)^i}{i!}}. \end{aligned} \tag{5.35}$$

Proof: We can express the conditional execution time of the time-based strategy in a similar way to that of the random checkpointing strategy as follows:

$$X^{(tc,f,h,l)}(n,v) = \begin{cases} X(n) + \sum_{j=1}^{K} H_j^{(l)} & \\ \qquad : \ \ if \ \ Y > X(n) + \sum_{j=1}^{K} H_j^{(l)} \ \ and \ \ X(n) \le v & \\ Y + R' + \tilde{X}^{(tc,f,h,l)}(n,p) & \\ \qquad : \ \ if \ \ (Y \le X(n) + \sum_{j=1}^{K} H_j^{(l)} \ \ and \ \ X(n) \le v) & \\ \qquad \ \ or \ \ (Y \le v + \sum_{j=1}^{K'} H_j^{(l)} + C \ \ and \ \ X(n) > v) & \\ v + \sum_{j=1}^{K'} H_j^{(l)} + C + X^{(tc,f,h,l)}(n-M,v) & \\ \qquad : \ \ if \ \ Y > v + \sum_{j=1}^{K'} H_j^{(l)} + C \ \ and \ \ X(n) > v & \end{cases} .$$

Here, M is the number of messages received during time period v. Then following the similar steps, we get (5.32) and (5.35). ■

(5.32) and (5.35) are recursive equations. It is difficult to derive the explicit expressions for the LST and the expectation of $X^{(tc,f,h,l)}(n,v)$; however, these equations can be analyzed numerically to evaluate the behavior of the time-based checkpointing strategy.

5.6 Comparisons and Discussions

In this section, we compare these three checkpointing strategies from different viewpoints. We introduce the average effectiveness, denoted by A, to evaluate the derived expected program execution times. The *average effectiveness* is the ratio between the expected program execution time without and with failures, handoffs, and checkpoints. It is expressed as

$$A = \frac{n}{\lambda \cdot E\left[X^{(c,f,h,l)}(n)\right]}. \tag{5.36}$$

The denominator of above equation expresses the expected number of messages received during program execution. From the average effectiveness, we can easily observe how much time is wasted due to events that interrupt the program execution. The parameter values illustrated below are selected for demonstration purposes [23, 44].

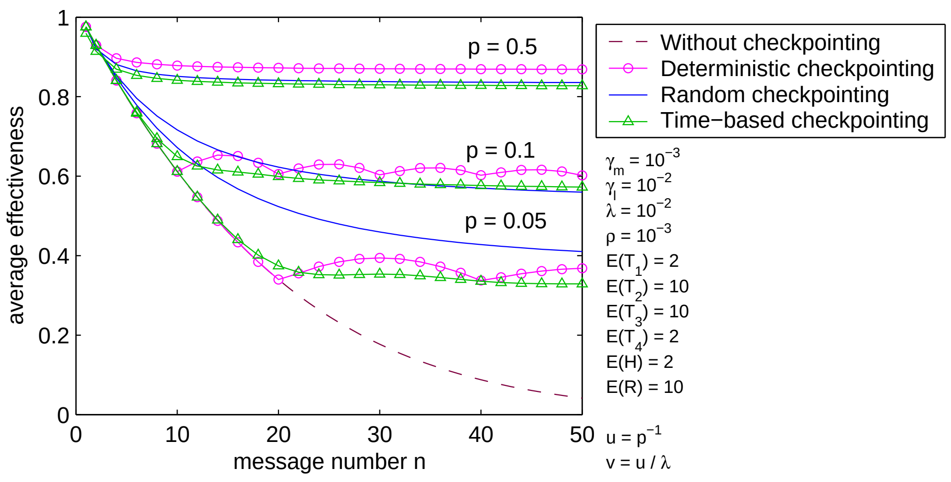


Figure 5.2: Average effectiveness vs. message number.

Figure 5.2 shows how the average effectiveness varies with the required number of computational messages[1] . The expectation of a r.v. which contains a geometric distribution with parameter p is p^{-1}, and the expected time to receive u messages with message arrival rate λ is u/λ. For comparison purposes, we group $u^{-1} = p = (v\lambda)^{-1}$ as a triplet and refer to these three parameters as *checkpointing frequencies*. When p tends to 1 (higher checkpointing frequencies), more checkpoints will be taken. The effectivenesses without checkpointing and with random checkpointing decrease monotonically with the message number n; however, the effectivenesses with deterministic and time-based checkpointing demonstrate fluctuating behaviors. For the deterministic checkpointing strategy, before n reaches u, its behavior is the same as that

[1]The average effectiveness is a discrete function of the message number n or u, but for clarity we plot the figure in continuous curves.

without checkpointing. After taking a checkpoint, the effectiveness may experience a sudden decrease (which depends on the cost and time to take and save a checkpoint) but it tends to increase as u' increases; however, when u' approaches u, the effectiveness decreases again, indicating that a new checkpoint should be introduced. The time-based checkpointing strategy introduces a more random characteristic with respect to the number of received messages than the deterministic checkpointing strategy does; therefore, its oscillating behavior is less obvious than that of the deterministic checkpointing strategy. From the curves for deterministic checkpointing, it is easily observed that checkpointing prevents the effectiveness from decreasing.

We also observe in Figure 5.2 that the random checkpointing will achieve higher effectiveness than the other two strategies when the checkpointing frequency decreases. As pointed out before, on average the number of checkpoints with random checkpointing is not less than that with deterministic checkpointing with the same checkpointing rate. When the checkpointing frequency is low, taking more checkpoints will exhibit positive effects on the effectiveness; on the other hand, when the checkpointing frequency is high, the checkpoint number is large enough, and no further benefit can be gained from checkpointing. As the random checkpointing introduces more checkpoints, it decreases the effectiveness correspondingly. Moreover, from this figure, we see that when n approaches 1, the average effectiveness with checkpointing may be less than that without checkpointing. As checkpointing itself brings an overhead, there should be a trade-off condition at which the expected program execution time with checkpointing will be less than that without checkpointing.

Figure 5.3 exhibits the variations of effectiveness with the message arrival rate λ. In this figure, we distinguish two time-based checkpointing strategies: adaptive and fixed. In the adaptive-time-based checkpointing strategy,

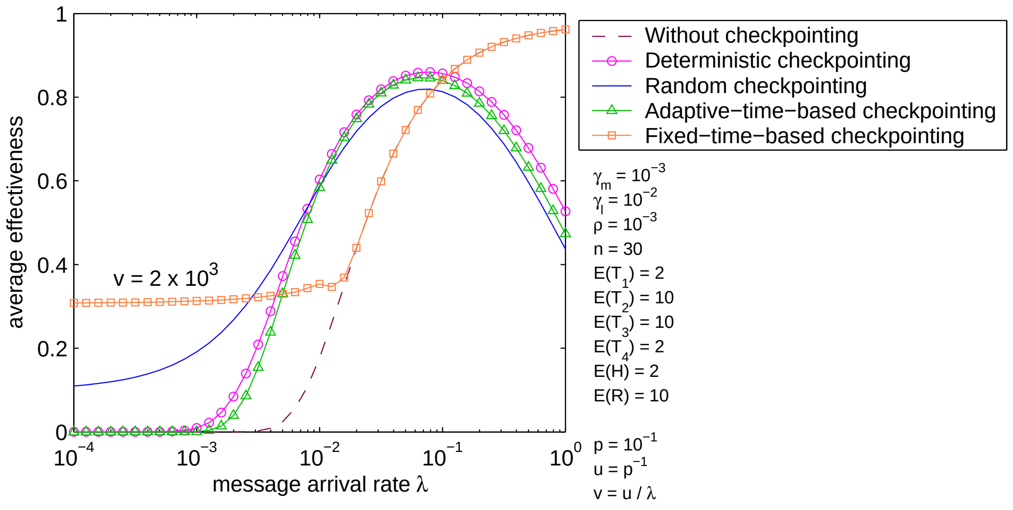


Figure 5.3: Average effectiveness vs. message arrival rate.

the checkpointing interval v changes with λ to keep the expected checkpointing frequency the same as those of deterministic and random checkpointing strategies. The fixed-time-based checkpointing implies that the interval v is constant and is independent of λ. Without checkpointing, the effectiveness increases with λ, as does that of the fixed-time-based checkpointing strategy. When the message arrival rate is low, checkpointing increases the effectiveness, especially with fixed-time-based and random checkpointing. However, the fixed-time-based checkpointing will be reduced to the strategy without checkpointing if the time to take a checkpoint, v, is greater than the time to complete the program when the message arrival rate is high. In contrast, for all the other three checkpointing strategies, the effectiveness increases first and then decreases. This is because the program will be completed with fewer failures when the message arrival rate is high. Consequently, checkpointing incurs too much overhead to offset against the gain it achieves. Under these conditions, we should take checkpoints less frequently to reduce the overhead.

Another observation is that the random checkpointing achieves better effectiveness when λ is small; however, as λ increases the deterministic checkpointing prevails.

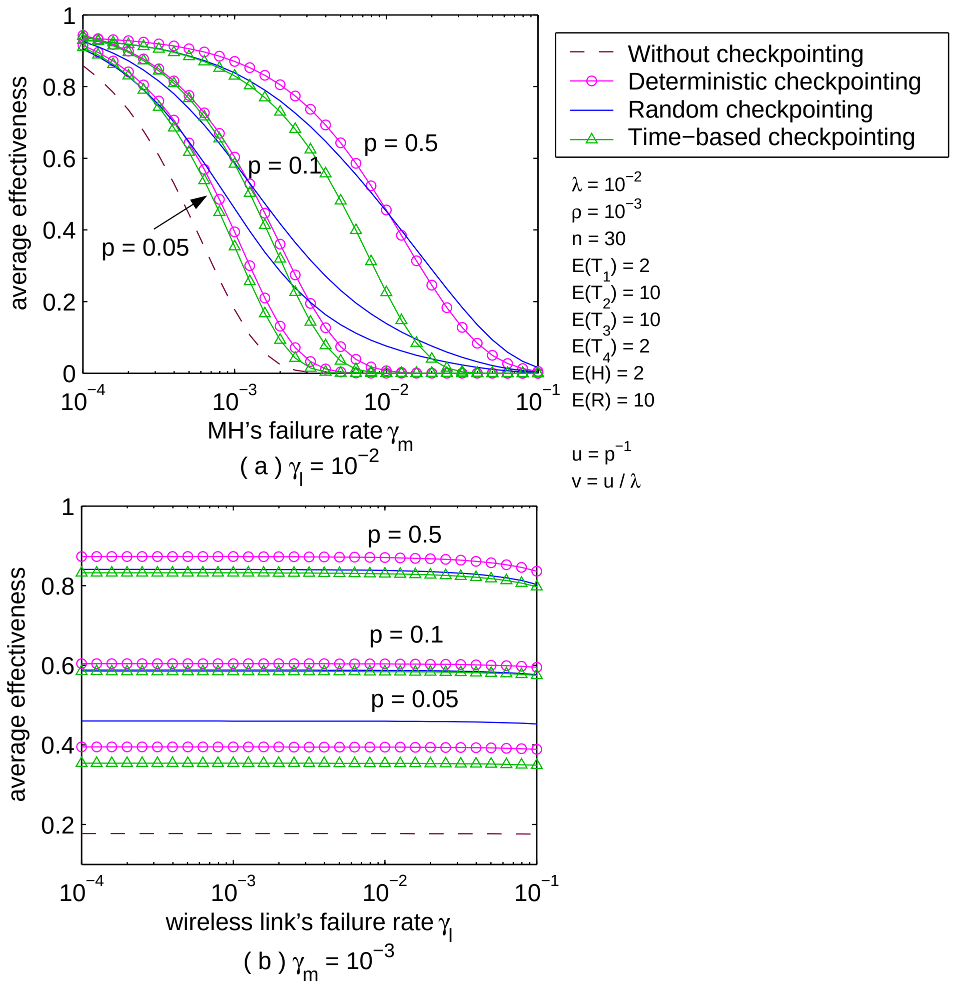


Figure 5.4: Average effectiveness vs. failure rate.

The variation of effectiveness with failure rates is shown in Figure 5.4. Irrespective of the different values of the checkpointing parameters for these three strategies, the effectiveness always decreases as the MH failure rate γ_m increases; however, this effect is much smaller with the wireless link failure rate

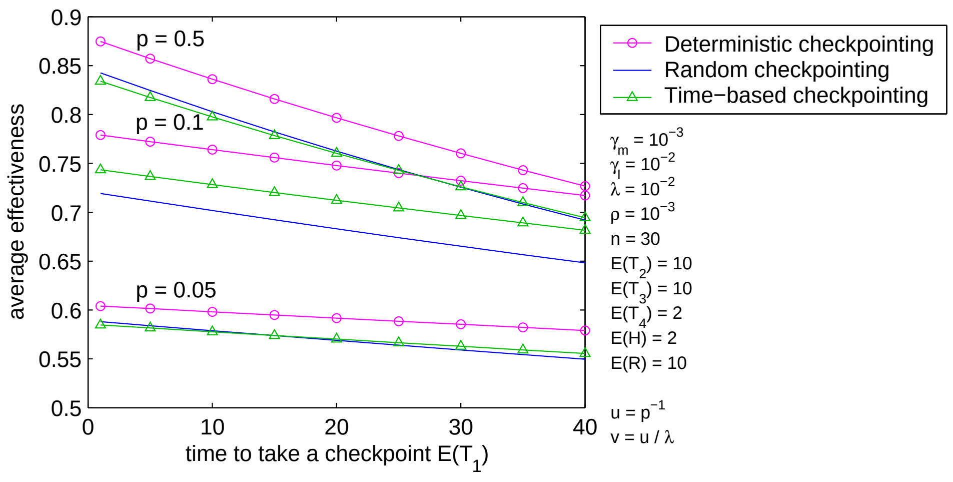


Figure 5.5: Average effectiveness vs. checkpoint taking time.

γ_l – in this case, the effectiveness remains almost constant. As seen in Figure 5.4(a), when γ_m is small, a higher checkpointing frequency may decrease the effectiveness; however, when γ_m is high enough, more frequent checkpointing affords higher effectiveness. Another phenomenon is that the deterministic and time-based checkpointing strategies achieve the higher effectivenesses when γ_m is small; as γ_m increases, the random checkpointing becomes the best.

Figure 5.5 shows how the effectiveness decreases as the checkpoint taking time T_1 increases with different checkpointing frequencies in these three checkpointing strategies, where T_1 is the overhead inherited from the checkpointing operation. This observation is also true for the other checkpointing overheads: T_2, T_3, and T_4. When T_1 is small, the effectiveness benefits from checkpointing; however, as T_1 becomes larger and larger, it brings more overhead, which eventually surpasses the benefit gained, resulting in less effectiveness. This figure also shows that the higher the checkpointing frequency, the faster the effectiveness decreases, which indicates that when it requires more time to take a checkpoint, the benefit of frequent checkpointing is more likely to be outweighed by the extra overhead.

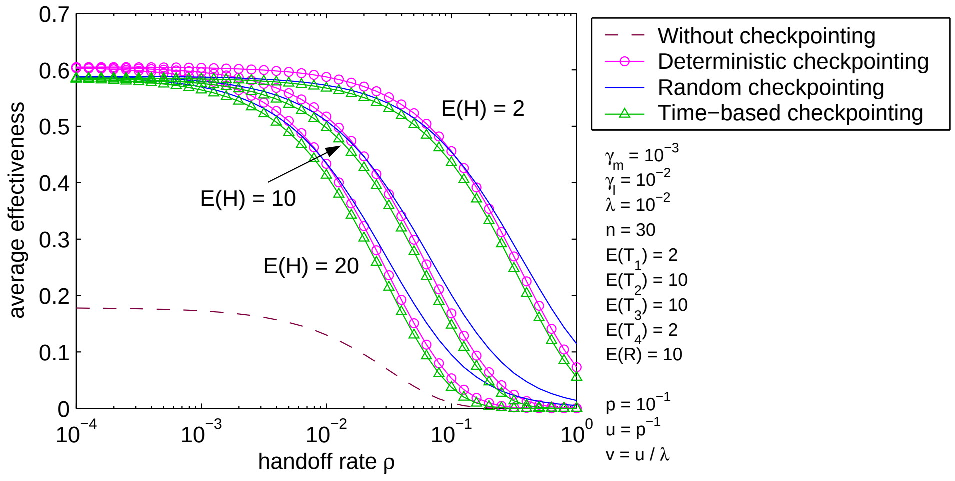


Figure 5.6: Average effectiveness vs. handoff rate.

The variations of effectiveness with the handoff rate ρ and the handoff time H are shown in Figure 5.6. The effectiveness decreases as ρ increases, whether we engage checkpointing or not. The three checkpointing strategies display little difference with the same ρ and H. Only when the handoff rate is high does the time required for handoff affect the average effectiveness significantly.

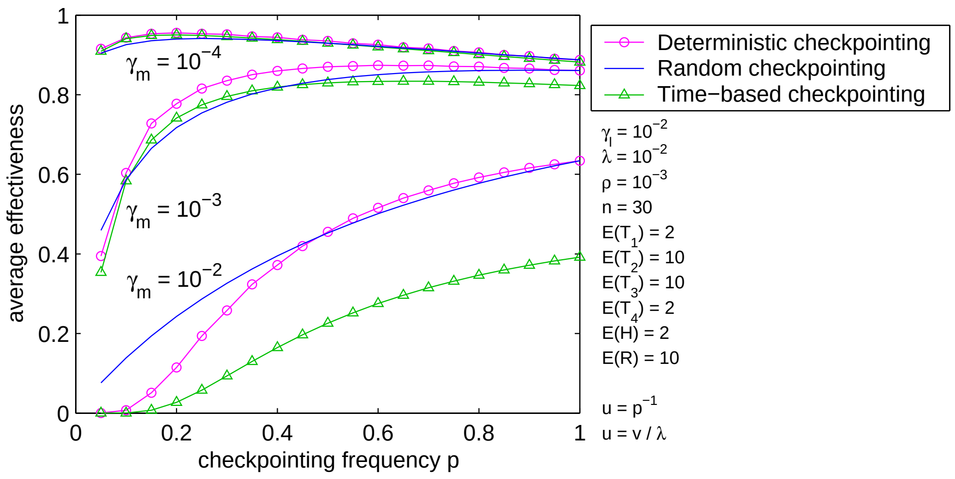


Figure 5.7: Optimal checkpointing frequency.

Figure 5.7 shows how to achieve the maximum effectiveness under different

MH's failure rates by adjusting the checkpointing frequency. The shapes of the curves change dramatically as γ_m varies. When the MH's failure rate is high enough, taking a checkpoint after receiving even single message in message-number-based checkpointing strategies, i.e., $u = p = 1$, is a good approach to assure high effectiveness. When γ_m is small, however, too high or too low checkpointing frequencies will result in low effectiveness, and there is an optimal checkpointing frequency that maximizes the effectiveness. Although it is difficult to derive explicit expressions for the optimal u, p, and v, these optimal values can be obtained by numerically solving the partial differentiation equations of the expected program execution times with respect to u, p, and v, respectively. Since u is an integer in the deterministic checkpointing strategy, we should choose one of the two nearest integers in this solution as the optimal u that yields the larger value of $E[X^{(dc,f,h,l)}(n,u)]$ [117]. If we cannot get solutions from these equations, the maximum effectiveness can be achieved by selecting $u = 1$, $p = 1$, and v as small as possible. It is also observed that a smaller γ_m usually requires fewer checkpoints to attain the maximum effectiveness.

Finally, from most of these figures, we can observe that the effectiveness with random checkpointing is more stable under varying parameter conditions than that with deterministic or time-based checkpointing.

5.7 Summary

In this chapter, we carry out the analyses of the program execution time with various checkpointing strategies in mobile wireless environments. The termination requirement for a program at MH is the number of computational messages received. We assume that MH and wireless link failure intervals, message arrival interval, and handoff interval are r.v.s with exponential distributions. Three checkpointing strategies, deterministic, random, and time-based checkpointing, are considered. We derive the LST of the c.d.f. of the total program execution time and its expectation. We show that the performance of random checkpointing approach is more stable against varying parameter conditions. Some trade-offs with respect to various parameters are also demonstrated. Based on our study, different checkpointing strategies, even including the strategy without checkpointing, can be engaged to achieve optimal performance under different mobile wireless network conditions. Nevertheless, the detailed measurement should be performed in order to deciding which strategy will be utilized.

□ **End of chapter.**

Chapter 6

Reliability Analysis for Various Communication Schemes

For the purpose of designing more reliable networks, we extend the traditional reliability analysis from wired networks to wireless networks with imperfect components. Wireless network systems, such as wireless CORBA, inherit the unique handoff characteristic which leads to different communication structures with various components and links. Therefore, the traditional definition of two-terminal reliability is not applicable anymore. We propose a new term, end-to-end mobile reliability, to integrate those different communication structures into one reliability metric, which includes not only failure parameters but also service parameters. Nevertheless, it should still be a monotonously decreasing function of time.

The end-to-end mobile reliability and its corresponding mean time to failure (MTTF) are evaluated quantitatively in different wireless communication schemes. To observe the gain in overall reliability improvement, the reliability importance of imperfect components are also evaluated. The results show that the failure parameters of different components take different effects on MTTF and reliability importance. With different expected working time of a system, the focus of reliability improvement should change from one component to another in order to receive the highest reliability gain. Furthermore, the number of engaged components during a communication state is more critical than the number of system states.

6.1 Definitions and Assumptions

In general, reliability is defined as the probability that a system performs its intended functions successfully for a given period of time under specified environmental conditions [111], and we refer the probability of successful communication between a source node and a target node as *two-terminal reliability* [118]. For two nodes to communicate with each other, there should be at least one operating path connecting them. An operating path indicates that all the intermediate nodes and links should be in the operation state: a node is operational if and only if it functions as intended with respect to specification; and a link is operational if and only if it allows communication from its source node to its sink node [128]. Because the two-terminal reliability problem in wired networks has been studied thoroughly in the literature, we assume that the intermediate nodes and wired links are always reliable, i.e., there will always be a reliable wired path between an AB and an SH, or between an AB and another AB. For the wireless part, an MH constructs only one wireless link with one AB, and it is associated with only one AB at a time, except during a handoff operation. Therefore, the communication path built on the top of wireless links is simple, and we also assume that wireless link failures are negligible. However, all the four components of wireless CORBA are failure-prone, and they may fail independently.

Based on the assumptions stated before, a successful communication between two nodes is defined as the condition when all the engaged nodes, including the source node and the target node, are in the operational state. As a result, the SH-SH reliability is the product of the two individual SHs' reliability. If one or both of the two terminals are MHs, the traditional two-terminal reliability metric cannot correctly describe the characteristic introduced by the handoff operation. As MHs move and perform handoff operations, the communication structures will vary with time t. Each communication structure can be regarded as a serial system composed of different types and numbers of engaged components. Additionally, the handoff operation induces that an MH's published address will be outdated, and a mechanism is needed to resolve the current location of the MH. Therefore, we propose a new term, end-to-end *Mobile Reliability (MR)* [26, 30], to address these unique cases in wireless environments. We define a system state, x, as the communication structure; therefore, x changes with time t. Furthermore, state z denotes the failure state, in which two terminals cannot communicate with each other. As a result, all communication states are up states. Let $\pi_x(t)$ denote the probability that the system is in state x at time t. The end-to-end MR at a generic time t, $MR(t)$, is given by

$$MR(t) = \sum_x \pi_x(t). \tag{6.1}$$

$MR(t)$ is a function composed not only of failure parameters but also of service parameters introduced by state probability $\pi_x(t)$. The SH-SH reliability can be treated as a special case in which the system contains only one communication structure, and $MR_{ss}(t) = \pi_x(t) = [R_{sh}(t)]^2$. Under the adopted assumptions, we can say that the MR is a generalization of the traditional two-terminal reliability. Accordingly, we define the corresponding end-to-end MTTF as

$$MTTF = \int_0^\infty MR(t)dt. \tag{6.2}$$

Although the proposed mobile reliability and performability [111] both combine service and failure parameters, actually they are different. Performability is utilized to measure gracefully degrading systems which may be able to survive the failure of one or more of their active components and continue to provide service at a reduced level. However, mobile reliability is a mere reliability metric for wireless mobile systems.

Four communication schemes will be generated if random communications occur between MH and SH, which are the SS scheme, the MS scheme, the SM scheme, and the MM scheme. In these notations, the former capital letter denotes the type of the source node, and the latter letter denotes the type of the target node, where M stands for MH, and S stands for SH.

During communications, an MH associates with an AB and exchanges messages with other nodes. As the MH moves, it will make handoffs and associate with a NAB. The sojourn time with an AB and the handoff completion time are assumed to be r.v.s which are exponentially distributed with parameters ρ and η, respectively. We assume that the component hazard rates are constant. That is, we model component failures as homogeneous Poisson processes, resulting in independent and exponential inter-failure arrivals [130]. The constant failure parameters for the four components of wireless CORBA, MH, AB, SH, and HLA, are γ_m, γ_a, γ_s, and γ_h, respectively. We utilize the exponential distribution as the service and failure distributions for modeling simplicity. As no repair is engaged in our evaluation, from the definition of MR in (6.1), the MR actually is the sojourn time at system communication states before the system moves to the failure state z, an absorption state. The derived properties of following discussions are only based on the monotonously decreasing characteristic of a reliability function with time, which is always the case; therefore, what the failure distribution really is should not affect the conclusions we will derive.

From the above definitions, we note that the end-to-end MR can be easily extended to include the reliability metrics of wired networks and wireless links if we add the two-terminal reliability of wired networks and the successful communication probability of wireless links into the calculation of MR. However, these extensions only trivially decrease the derived value of MR, but do not change the properties of MR; for simplicity we omit them in this chapter.

6.2 End-to-end MR and MTTF Analysis

Different communication schemes engage various types and numbers of components which result in different end-to-end MRs and MTTFs. The SS scheme is trivial, and its MR has been derived in the last section, i.e., $MR_{ss}(t) = [R_{sh}(t)]^2$. Therefore, we will discuss the remaining MS, SM, and MM schemes in the following three subsections separately.

6.2.1 The MS (MH-SH) Scheme

The MS scheme is a communication scheme in which an MH initiates communications with an SH. Initially, the MH sends requests over a wireless link, then the associated AB relays the request messages to the target SH through wired paths. After a random sojourn time in the current AB, the MH may perform a handoff during which two ABs are engaged. The system states are thus shown in Figure 6.1, in which solid lines denote wired paths while dashed lines denote wireless links. State a is a normal communication state, and state b is a handoff state in which the MH moves from AB$_1$ to AB$_2$. The handoff may be network initiated or terminal initiated [100]; however, the engaged nodes and links are the same. The MH should create two different wireless links with two

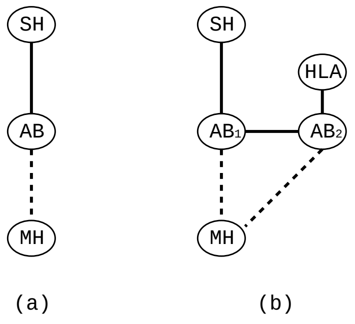
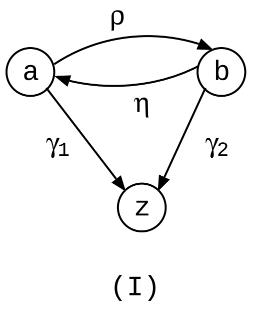


Figure 6.1: System states in the MS scheme: (a) normal communication; (b) handoff procedure.



$$\boxed{\begin{aligned}\gamma_1 &= \gamma_m+\gamma_a+\gamma_s\\ \gamma_2 &= \gamma_m+2\gamma_a+\gamma_s+\gamma_h\end{aligned}}$$

Figure 6.2: Markov model for the MS scheme.

ABs, and these two ABs should inform each other about the handoff progress. During handoff, the NAB, AB_2, should invoke the *location_update* operation at the MH's HLA to inform it that the MH has changed its associated AB. We may exclude the HLA from state b if we employ a simple invocation retry strategy, and the MH's location in the HLA will eventually be updated no matter whether the HLA works or not during the handoff. This is a simple extension to improve the system's reliability. After the handoff, the system returns to state a for normal communications. Figure 6.2(I) shows the Markov model of the system state transition, where ρ is the handoff rate, η is the handoff completion rate, and γ_1 and γ_2 are the failure rates at states a and b, respectively.

The Kolmogorov differential equations for this Markov model should be

$$\begin{cases} \frac{d\pi_z(t)}{dt} = \gamma_1\pi_a(t) + \gamma_2\pi_b(t) \\ \frac{d\pi_a(t)}{dt} = -(\gamma_1+\rho)\pi_a(t) + \eta\pi_b(t) \\ \frac{d\pi_b(t)}{dt} = \rho\pi_a(t) - (\gamma_2+\eta)\pi_b(t) \end{cases}.$$

From the above equations, the probabilities of the system in states a, b, and z at time t can be solved; however, the expressions are not very concise, so we omit them here. But the MTTF for the MS scheme can be shown with a concise equation as follows:

$$MTTF_{ms} = \frac{\gamma_2+\rho+\eta}{\gamma_1\gamma_2+\gamma_1\eta+\gamma_2\rho}. \tag{6.3}$$

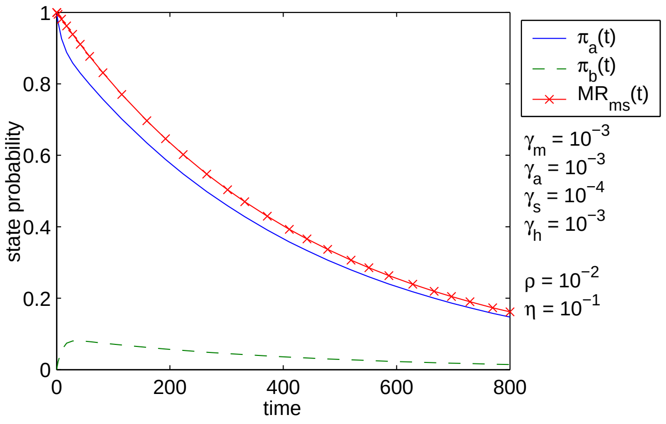


Figure 6.3: State probabilities and reliability of the MS scheme.

One realization of the end-to-end MR of the MS scheme, $MR_{ms}(t)$, is shown in Figure 6.3, under the assumption that the initial state of the Markov model is a. Different types of components experience different levels of failures. SHs are generally more reliable than MHs or ABs; therefore, we let $\gamma_m = \gamma_a = 10^{-3}$ and $\gamma_s = 10^{-4}$ [23, 105]. We select the specific values of parameters for demonstrating the proposed end-to-end MR, whereas these values are set at a reasonable and comparable level. As expected, the probability of the system in state a is much greater than that in state b as the handoff procedure is completed very quickly, resulting in a case that the probability of state a contributes much more to the MR than that of state b. Although $\pi_b(t)$ increases first and then decreases, the end-to-end MR is still a monotonously decreasing function of time t.

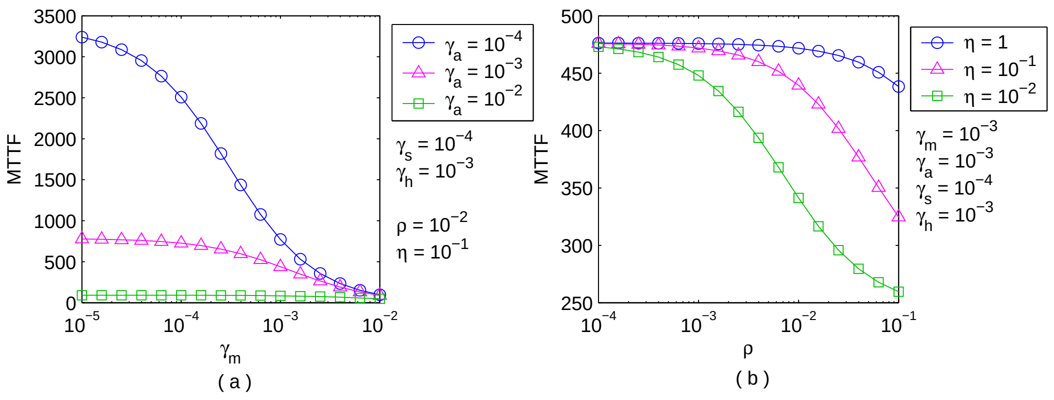


Figure 6.4: End-to-end MTTF of the MS scheme: (a) failure parameters γ_m and γ_a; (b) service parameters ρ and η.

Figure 6.4 shows the end-to-end MTTF as a function of failure and service parameters. The more reliable the components are, the longer the MTTF is. However, the improvement gain (in terms of the MTTF) is reduced with the decrease in the failure parameters, γ_m and γ_a, beyond a certain threshold, which can be observed from Figure 6.4(a). Such diminishing gain should be carefully considered against the cost of increasing component reliabilities beyond a limit [130]. This result is also applied to parameter γ_s. From (6.3), we see that γ_m and γ_s produce the same effect on $MR_{ms}(t)$, and little difference exists between γ_m and γ_a when $\pi_b(t)$ is much smaller than $\pi_a(t)$. This means that these three components are critical to successful system communications. The change of the failure rate of the HLA, γ_h, dose not make the MTTF demonstrate obvious variations, which implies that any gain by improving the reliability of the HLA will be small. This is because γ_h is only taken into consideration in state b, which does not contribute much to the MR. Figure 6.4(b) shows that when ρ is high, the MTTF increases with η dramatically; however, when ρ is low, the MTTF varies little with η. This indicates that when the handoff happens frequently, the time spent in the handoff period is very critical to the MTTF, because the reliability is clearly lower in the handoff state b than in the normal state a. When ρ is low, however, the contribution of state b is small, leading to little change of the MTTF with η. To achieve a higher MR, then, MHs experiencing high handoff rates should complete the handoff operation as fast as they can.

6.2.2 The SM (SH-MH) Scheme

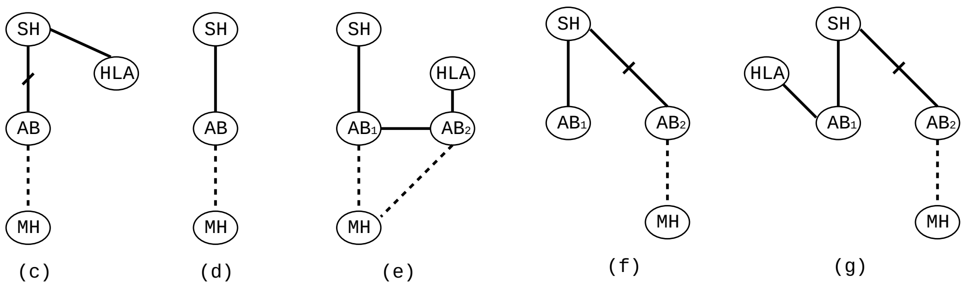
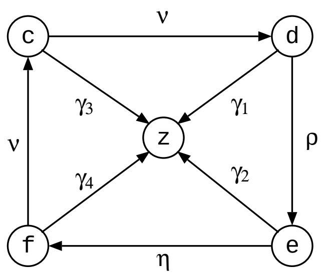
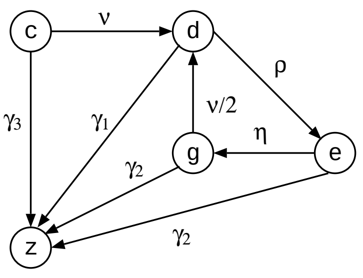


Figure 6.5: System states in the SM scheme: (c) location-querying; (d) normal communication; (e) handoff procedure; (f and g) location-forwarding.

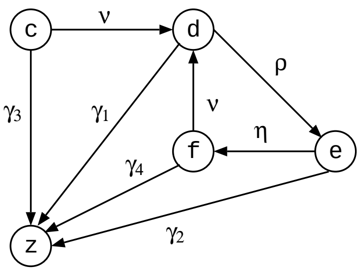
In the SM scheme, an SH initiates communications with an MH. The characteristic of an MH is its movement, which introduces a mechanism to locate its current AB. The location-lookup mechanism complicates the system states, as shown in Figure 6.5. We know that an object on an MH publishes its Mobile Interoperable Object Reference (MIOR) with the address of its resided MH's HLA. When an SH first invokes an object on an MH with the originally



(II) LF_HLA



(III) LF_QHLA



(IV) LF_AB

$$\boxed{\begin{aligned}\gamma_1 &= \gamma_m+\gamma_a+\gamma_s\\ \gamma_2 &= \gamma_m+2\gamma_a+\gamma_s+\gamma_h\\ \gamma_3 &= \gamma_m+\gamma_a+\gamma_s+\gamma_h\\ \gamma_4 &= \gamma_m+2\gamma_a+\gamma_s\end{aligned}}$$

Figure 6.6: Markov models for the SM scheme.

published MIOR, the request message will be sent to the HLA indicated according to the address specified in the MIOR, and the HLA will send back a GIOP reply message with status *LOCATION_FORWARD*. This reply message carries a renewed MIOR containing the address of an AB with which the HLA believes the MH is currently associated [100]. This is state c, in which the solid line with slash indicates that the SH has not yet created a communication path with the AB; however, it is tending to construct such path with the AB. The time spent in this state is also assumed to be an exponentially distributed r.v. with parameter ν. The received LOCATION_FORWARD message directs the SH to reissue the request to the AB, and then the AB forwards the message to the MH. This is state d. The system will stay in this normal communication state until the MH moves out of the coverage area of the current AB. State e is the handoff state. As the SH does not know whether or not its target MH has experienced a handoff, it will still send its subsequent requests to its known AB as normal despite the movement of the MH. However, when the AB receives a request and finds that it has broken its link with the targeted MH, it will reply with a message whose status is also set to be LOCATION_FORWARD. There exist two ways to construct this reply message by replacing the address part in the IIOP profile of the MIOR with different addresses. One is the address of the MH's HLA, and the other is the address of the MH's current AB. We denote the former location-forwarding approach as *LF_HLA* and the latter as *LF_QHLA*. Figures 6.6(II) and (III) show their corresponding Markov models, respectively.

In the LF_HLA approach, after handoff, the system moves from state e to state f then to state c, resending the request to the HLA; however, the LF_QHLA approach changes state e to state g during which the AB queries the current location of the MH from its HLA. With the address of the NAB, the SH can reissue the request directly to the NAB, which results in the system transferring from state g directly to state d. The time spent in state f is also assumed to be an exponentially distributed r.v. with parameter ν because it functions as location-forwarding the same as state c. However, the transition rate from state g to state d should be different as state g engages one more operation than state f does. Here we set it to be $\nu/2$. The specific relationship between these two rates are not important here, but these two rates should not be independent. One more location-forwarding approach, denoted as *LF_AB*, could be engaged. Actually, the OAB knows to which AB the MH moves away from itself. If the MH does not leave the NAB when the OAB receives a request on this MH, the information in the OAB about the location of the MH is up-to-date, and it could be employed to construct the reply message. With this approach, the SH can still resend the request to the current AB; however, the location-querying operation in the LF_QHLA approach is removed. Its corresponding Markov model is shown in Figure 6.6(IV).

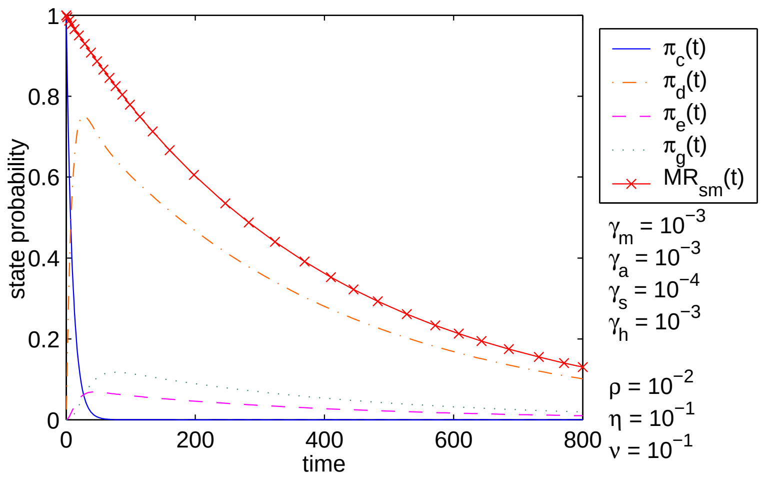


Figure 6.7: State probabilities and reliability of the SM scheme.

The symbolic expression of the probabilities of the system in different states at time t are difficult to be derived. Therefore, we utilize a numerical approach here to express their variations with time t. Figure 6.7 shows one realization for the LF_QHLA approach. The curve shapes of the other two approaches are similar to this one. We observe that the contribution of state c to the MR decreases quickly as time moves on, because the probability of state c diminishes quickly. States d, e, and g exhibit similar behaviors in MR as all these three states show similar curve shapes in state probability.

Figure 6.8 shows the differences in MR among these three location-forwarding approaches with various handoff rates. It is observed that the proposed LF_AB approach achieves the highest MR because it engages the least number of components and finishes the location-forwarding procedure the most quickly. However, these three approaches tend to behave the same when the handoff rate decreases. Another observation is that the LF_HLA approach is superior to the LF_QHLA approach. Although the LF_QHLA combines two states f

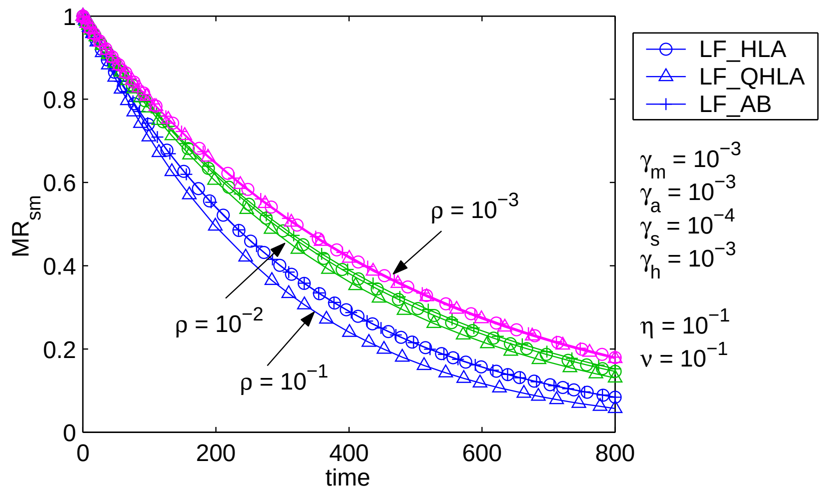


Figure 6.8: Reliability with location-forwarding strategies in the SM scheme.

and c into one state g, at the same time it introduces one more component, the HLA, into state g. This indicates that the number of engaged components during a communication state is more critical than the number of states.

How the MTTF varies with different parameters are shown in Figures 6.9 and 6.10. We note that γ_m, γ_a, and γ_s produce similar effects on the MTTF of the SM scheme as those of the MS scheme. In Figure 6.9(b), the decrease of γ_h also demonstrates an increase on the MTTF. This implies that the HLA plays a more important role in the SM scheme than in the MS scheme, although the improvement on the reliability of the HLA still achieves little increase on the MTTF compared with the improvement on other components. Finally, the increase in η with a high handoff rate shows smaller increase in the MTTF than that produced in the MS scheme. This may be explained as only two communication states in the MS scheme; however, more communication states are engaged in the SM scheme. The increase in the communication states reduces the effect of parameter η due to the introduction of one more parameter ν, as shown in Figure 6.10(b). It exhibits that the the location-forwarding process should be done as quickly as possible despite of the value of ρ.

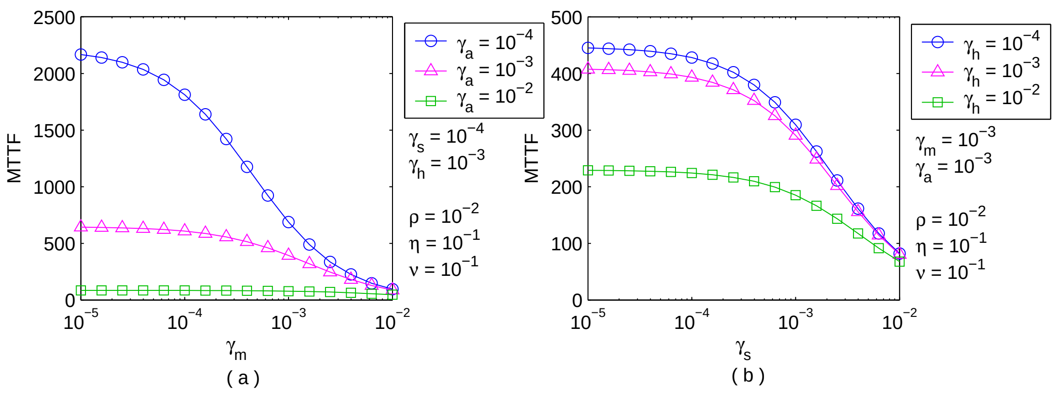


Figure 6.9: End-to-end MTTF of the SM scheme vs. failure parameters: (a) γ_m and γ_a; (b) γ_s and γ_h.

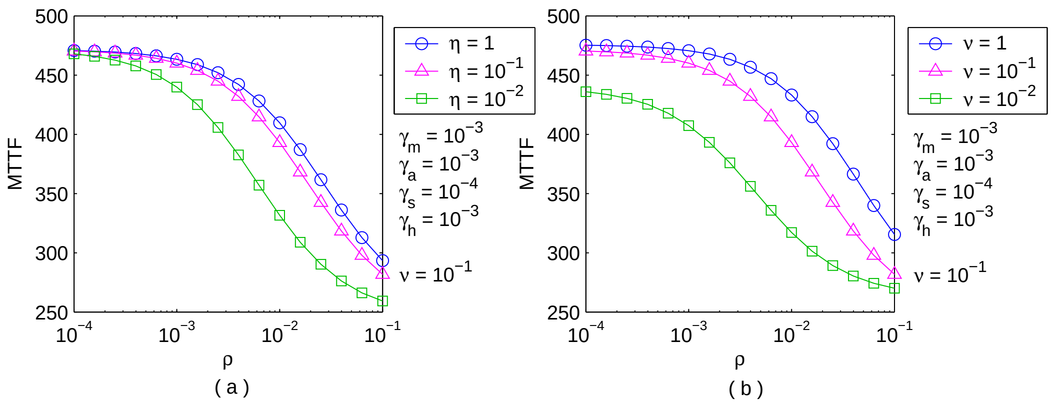


Figure 6.10: End-to-end MTTF of the SM scheme vs. service parameters: (a) ρ and η; (b) ρ and ν.

We have observed that the MH and the AB behave almost the same in the improvement gain in terms of the MTTF in schemes MS and SM. Now we evaluate them from another point of view to see whether this result will be changed or not. We define *time-dependent RI* with respect to the MR to identify the relative importance of each component in a system. The time-dependent RI, $I_{R_i}(t)$, of component i, $i = mh$, ab, sh, or hla, is given by

$$I_{R_i}(t) = \frac{\partial MR(t)}{\partial R_i(t)}. \tag{6.4}$$

Applying (6.4) in the SM scheme, we show the results in Figure 6.11. When the handoff rate is relatively high (Figure 6.11(a) and (b)), the RI of the AB increases first and then decreases, indicating the contribution of state e, f, or g is high. If the AB and the MH experience the same failure rate (Figure 6.11(a)), the AB always gets the higher RI than the MH does. On the other hand, if the AB is more reliable than the MH (Figure 6.11(b)), the AB gets the higher RI initially, and then the MH gets the higher RI; otherwise, the MH always gets the higher RI with a lower handoff rate (Figure 6.11(d)).

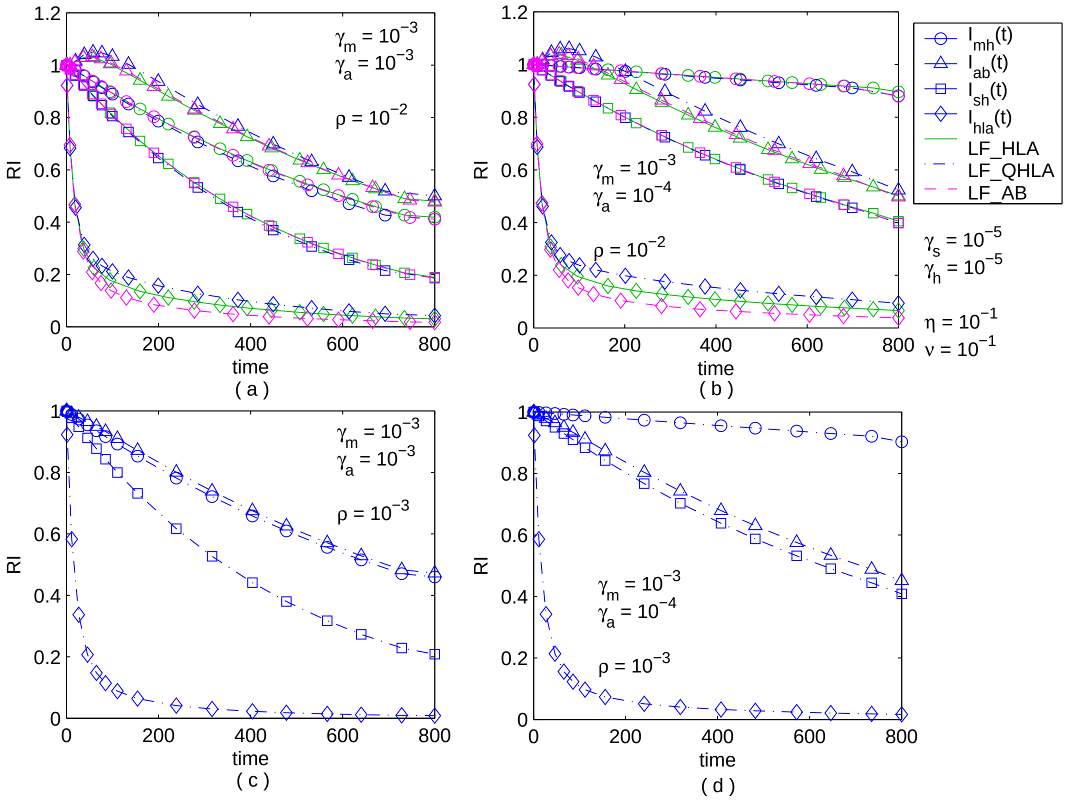


Figure 6.11: RI of the SM scheme: (a) same failure rate and high handoff rate; (b) different failure rates and high handoff rate; (c) same failure rate and low handoff rate; (d) different failure rates and low handoff rate.

These observations show that the relative RIs of different components may vary with the intended working time of the system and with the failure and service parameters. The RIs of HLA and AB in the LF_QHLA approach are higher than those in the LF_HLA and LF_AB approaches. This is due to the larger sojourn probability in state g which incorporates both AB and HLA. We compare the difference between Figure 6.11(a) and (c), in which the AB and the MH inherit the same failure rate; so do the SH and the HLA. We only show the result of the LF_QHLA approach, as the other two behave almost the same. Even when AB and MH experience the same failure rate, the difference between their RIs is relatively large when the handoff rate is relatively high; however, they get almost the same RI when the handoff rate is relatively low. The SH gets the higher RI than the HLA does despite the handoff rate. All these are induced by the probabilities of different system states in which each component engages. When the handoff rate is high, the system achieves greater probabilities in state e, f, or g, in which two ABs are employed. Therefore, the RI of the AB will be higher than that of the MH. The SH is present in each system state, but the HLA does not appear in state c, which is the most important state. Obviously then, the RI of the SH should always be higher than that of the HLA.

6.2.3 The MM (MH-MH) Scheme

The system becomes more complicated in the MM scheme as both MHs may undergo handoffs, and the following location-forwarding approaches also increase the system states. Its system states are shown in Figure 6.12, in which the failure state z is omitted for simplicity as each system communication state has a transition to state z. At first, the system is in state h, in which MH_1 is the invocation initiator, and MH_2 is the receiver. When MH_1 sends a request with the MIOR of an object on MH_2, its associated AB, AB_1, needs a location-forwarding approach to resolve the address of AB_2 in which MH_2 resides. Note that the renewed MIOR is only kept by AB_1, and MH_1 still

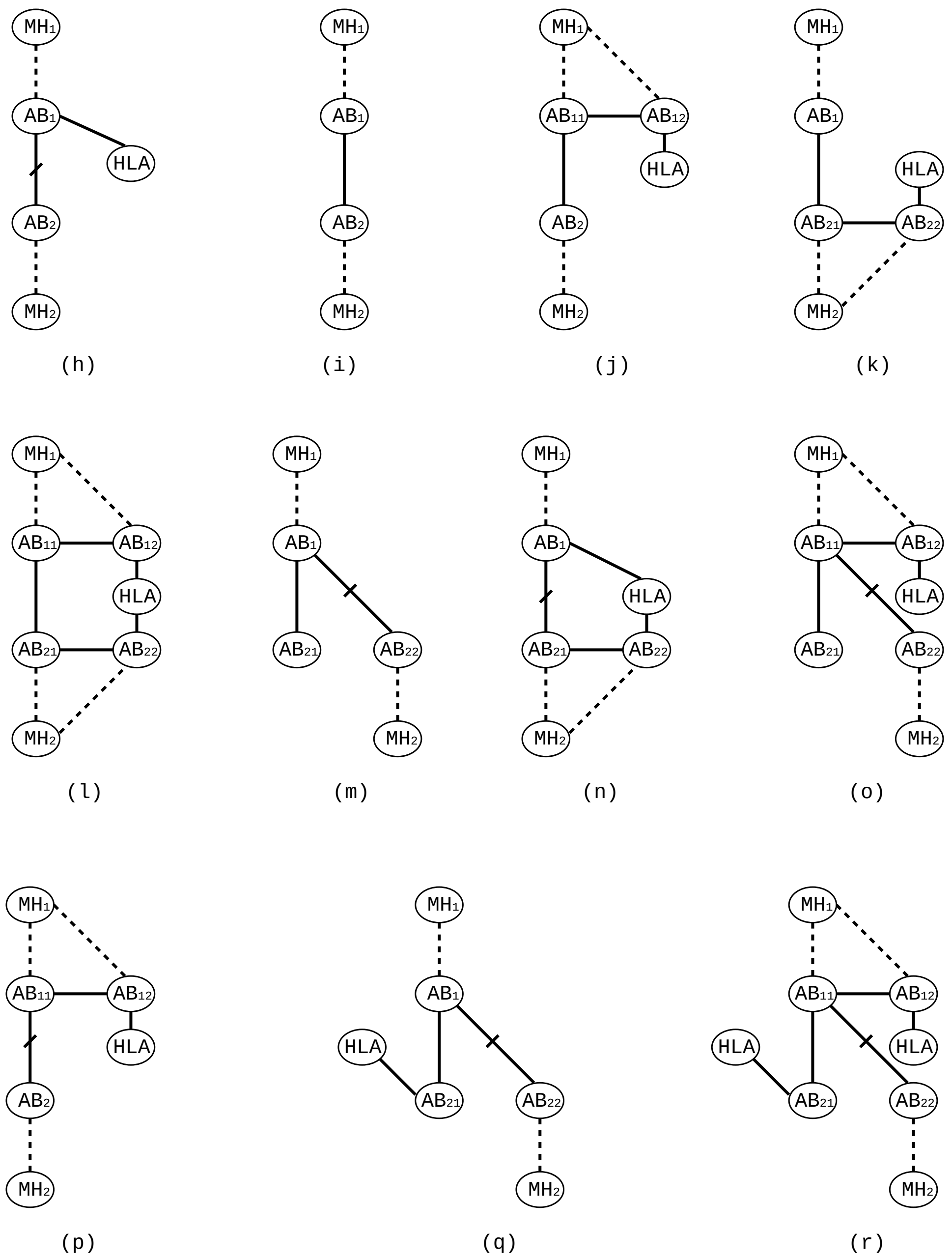


Figure 6.12: System states in the MM scheme: (h) location-querying; (i) normal communication; (j) MH_1 in handoff; (k) MH_2 in handoff; (l) both MH_1 and MH_2 in handoff; (m and q) location-forwarding; (n) location-querying and MH_2 in handoff; (o and r) location-forwarding and MH_1 in handoff; and (p) location-querying and MH_1 in handoff.

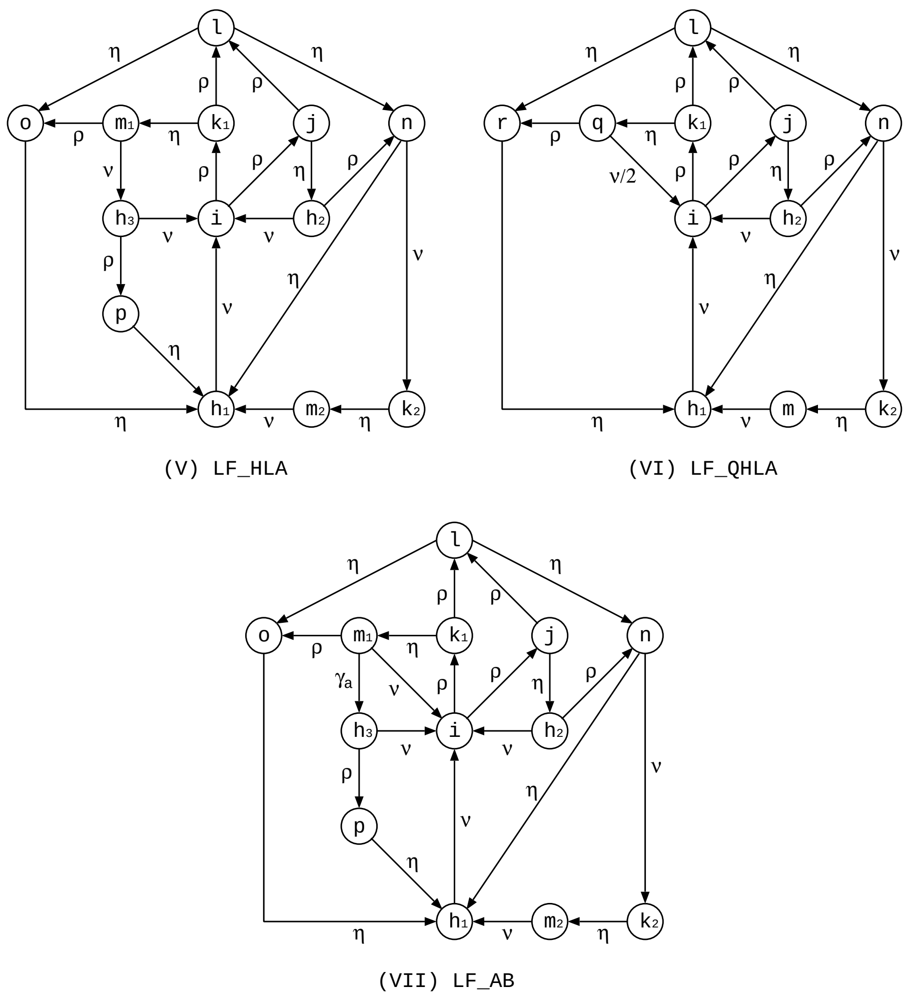


Figure 6.13: Markov models for the MM scheme (transitions to the failure state z is omitted).

keeps the original MIOR. After this step, AB_1 creates a communication path with AB_2, and then MH_1 sends messages to and receives messages from MH_2 through AB_1 and AB_2, without the interaction with the HLA. This is state i. States j and k denote the system states in which only one MH is in handoff. There exists a probability that both MH are in handoff, shown as state l. Here we assume that these two MHs share one HLA and do not reside within the same cell of an AB. These assumptions are reasonable because we could regard the derived results as the lower bounds, and the difference is small. Following state j, there are two possible transitions: one is to state l, and the other is to state h. State j cannot directly transit to state i, because after MH_1 moves to a NAB, this NAB does not contain any information about where MH_2 is, and thus it needs undergo state h to resolve the address of MH_2. There are also three location-forwarding approaches after the handoff completion of MH_2 in the MM scheme, denoted as the same as in the SM scheme. For the LF_HLA approach, state m is the location-forwarding process after the handoff completion of MH_2 in state k. No matter which MH finishes its handoff first in state l, the system also enters the location-forwarding state, state n or o; however, entering which location-forwarding state depends on which MH completes its handoff earlier. From it we know that the location-forwarding approaches after the handoffs of MH_1 and MH_2 are different.

The corresponding Markov model for the LF_HLA approach is shown as Figure 6.13(V), in which h_1, h_2, and h_3 represent the same communication structure as h while they represent different system states, the same denotations for k and m. One more assumption has been made to draw this Markov model: before an MH makes another handoff, the location-forwarding process for its last handoff should have been finished. This assumption avoids creating an AB list in which the MH is moving to the header AB of this list while the location-forwarding procedure is being processed in the tail AB of this list. It is also feasible because the handoff rate is much less than the location-forwarding rate. With the LF_QHLA approach (Figure 6.13(VI)), state q replaces state m, and it transits directly to state i instead of through state h_3. Correspondingly, state o is replaced by state r. Figure 6.13(VII) is the Markov model for the LF_AB approach. We could make this approach more reliable by adding a transition from state m_1 to h_3 with rate γ_a. Because AB_{21} functions as an HLA when the system is in state m, the HLA could be treated as a hot standby component to replace AB_{21} when AB_{21} fails, even though the failure rates may be different between these two components. When AB_{21} fails to reply to a request, the AB_1 may reissue the request to the HLA to get the up-to-date MIOR. The wireless CORBA specification requires that only the ORB used to implement HLA and AB need to know the mobility of MH, therefore this proposed request-retry could only be employed in the MM scheme while it is not suitable for the SM scheme, otherwise the ORB in SH needs to be aware of terminal mobility.

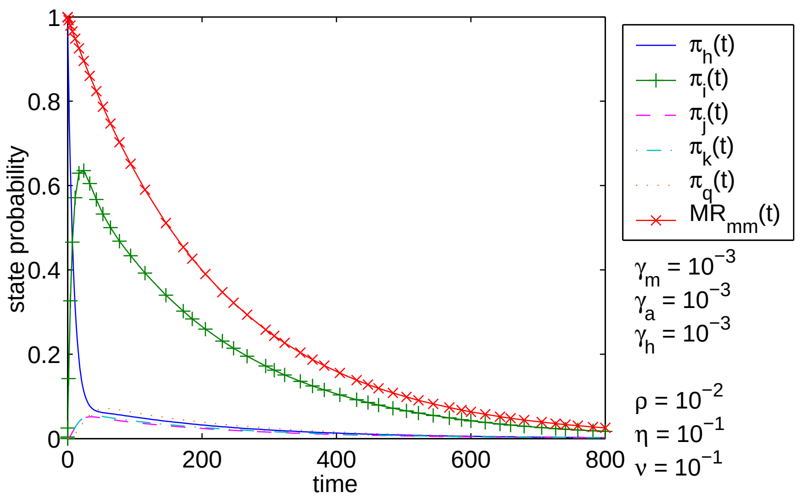


Figure 6.14: State probabilities and reliability of the MM scheme.

The variations of the probabilities of system states with time t for the LF_QHLA approach are shown in Figure 6.14 by employing that the system is initially in state h with probability 1. The probabilities of states j, k, and q are on one level of magnitude, and the probabilities of states l, m, n, and r are on another level of magnitude; however, all these states employ the same curve shape. The effects of different parameters on the MTTF and the RI in the MM scheme are very similar to those analyzed and presented in the SM scheme, and thus are not included here.

6.2.4 General End-to-end MTTF

We have discussed the end-to-end MTTF with specific sender-receiver pairs in four communication schemes so far. Now we turn our attention to the general end-to-end MTTF of a wireless communication system which includes n_{mh} MH and n_{sh} SH. If each MH or SH has the same probability to initiate a communication, then the general end-to-end MTTF can be expressed as

$$\begin{aligned} MTTF &= \frac{1}{2n_{mh}n_{sh} + \binom{n_{mh}}{2} + \binom{n_{sh}}{2}} \cdot \Big[n_{mh}n_{sh} \cdot MTTF_{ms} + n_{sh}n_{mh} \cdot MTTF_{sm} \\ &+ \binom{n_{mh}}{2} \cdot MTTF_{mm} + \binom{n_{sh}}{2} \cdot MTTF_{ss} \Big], \end{aligned} \tag{6.5}$$

in which we assume that all MHs share a common HLA.

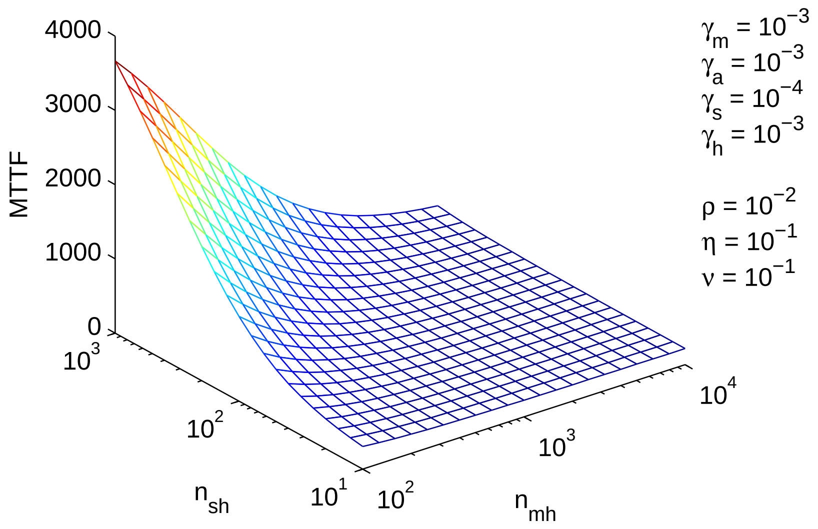


Figure 6.15: General end-to-end MTTF vs. number of components.

Figure 6.15 shows how the general end-to-end MTTF varies with the number of nodes, in which the LF_QHLA approach is utilized for the SM and MM schemes. As expected, the MTTF decreases with the number of MH; however, it increases with the number of SH. The $MTTF_{sm}$ or $MTTF_{ms}$ is larger than the $MTTF_{mm}$ under the same parameter values as more components are engaged in the MM scheme. If the number of SH increases, an MH will communicate with an SH more probably; then the MTTF will become larger. The number of AB may also affect the MTTF because the MH needs AB to relay messages. According to our definition of the general end-to-end MTTF, however, the number of AB has no effect on it.

6.3 Summary

Four communication schemes have been discussed in this chapter. No handoff operation is engaged in the SS scheme; as a result $MR_{ss}(t)$ is the traditional two-terminal reliability. For all other three schemes, MS, SM, and MM, the unique feature of wireless networks, handoff, is integrated into the mobile reliability. Quantitative measurements reveal that the handoff and location-forwarding procedures should be completed as soon as possible to improve the MTTF. Moreover, the RI of different components should be determined with specific failure and service parameters. Finally, the number of engaged components during a communication state is more critical than the number of system states.

For simplicity, we assume that the wired and wireless communication links are perfect and omit them in the reliability analysis. If these two are engaged into the proposed end-to-end mobile reliability, it can give a more detailed and complete reliability assessment of a wireless network system. Our quantitative measurements are conducted as an example with the assumption that the failure and service rate are constant; however, in practice, failure and service processes may follow other distributions. After all, our investigation provides an initial yet overall approach to measure the reliability of wireless networks. Although our analysis is conducted on wireless CORBA platforms, it is easily extensible to generic wireless network systems.

□ **End of chapter.**

Chapter 7

Sensibility-Based Sleeping Configuration in Sensor Networks

For wireless sensor networks, developing a localized configuration protocol for sensor sleeping is an effective approach to obtain a long network lifetime without sacrificing crucial aspects of quality of service, such as area coverage, sensing reliability, and network connectivity, etc. Different types of sensors generally have widely various theoretical and physical characteristics; therefore, different sensing models should be constructed to capture various sensing characteristics, which depend on the specific sensing device and the deployment environment. In this chapter, two sensing models are investigated: Boolean sensing model (BSM) and collaborative sensing model (CSM). Two sleeping candidate conditions are identified for the BSM. Firstly, we propose minimum partial arc-coverage (MPAC) to exploit the arc-coverage information provided by one-hop neighbors. Each sensor engages different sensing radii, and the deployment region preserves its k-coverage requirement. Secondly, a sensor evaluates the coverage of Voronoi vertices constructed by its one-hop neighbors, by which it can decide whether itself is sleeping-eligible. For the CSM, we exploit the cooperation between neighboring sensors by evaluating the proposed neighboring-sensor field sensibility (NSFS). After scheduling sleeping-eligible sensors, the constructed network remains connected in the presence of sensor's location error.

7.1 Assumptions and Problem Formulation

To configure a sensor to sleep while preserving area coverage in a decentralized network environment, we should answer three fundamental questions: when we can assert that an area is covered by a set of sensors; what each sensor's responsibility is in providing area coverage; and whether its sleeping will produce any reduction on covered area.

Some general assumptions are introduced to help us address these three questions. We assume that each sensor N_i knows its own location (x_i, y_i) [126, 133, 136, 143, 145], which can be obtained from the GPS or other localization systems [57]. Initially, we assume that the obtained location information is accurate; however, this assumption will be relaxed in Subsection 7.2.4. The sensors discussed in this chapter are deployed in a two-dimensional constrained Euclidean plane; however, the argument can be easily extended to a three-dimensional space. Sensors can communicate directly with their neighboring sensors within radius cr.

Definition 1 The one-hop communication neighbor set of sensor N_i is defined by

$$N(i) = \{N_j \in \Omega \mid d(N_i, N_j) \leq cr,\ j \neq i\}, \tag{7.1}$$

where Ω is the sensor set in a deployment region Φ and $d(N_i, N_j)$ denotes the Euclidean distance between sensors N_i and N_j.

7.1.1 Boolean Sensing Model

The Boolean sensing model (BSM) assumes that the sensing area of a sensor N_i is the disk with a radius sr_i centered at the location of the sensor itself [58, 126, 133, 136, 143], and we call its sensing area the *sensing disk*, denoted as Ψ_i. Thus,

Definition 2 A measuring point y in a constrained deployment region Φ is defined as being covered if there is at least one sensor N_i whose distance to point y is less than its sensing radius sr_i, i.e.,

$$\exists N_i \in \Omega,\ d(N_i, y) < sr_i. \tag{7.2}$$

In addition, we call the border of a sensor's sensing disk the *sensing perimeter*. If every measuring point in a deployment region is covered, we say that the deployment region is covered. Note that a sensor's sensing radius sr_i is different with its communication radius cr because different devices are involved [45]. As communication is usually bi-directional, the communication radius of all sensors are set the same. Although most sensor networks use homogeneous sensors with the same type [55, 145], sensors may still employ different sensing radii due to manufacturing deviation.

With the BSM, a sensor can ensure to detect an event occurring in its sensing disk by itself; however, if an event is outside its sensing radius, it has no way to perceive the event occurrence. In general, the ability of a sensor to detect the occurrence of an event of interest at a certain point degrades as the distance between the sensor and the point increases. To capture such characteristic, some authors also adopted a collaborative sensing model [79, 82, 90].

7.1.2 Collaborative Sensing Model

Definition 3 In a collaborative sensing model (CSM), the sensibility of a sensor N_i for an event occurring at an arbitrary measuring point y is defined by

$$s(N_i, y) = \frac{\alpha}{[d(N_i, y)]^{\beta}}, \tag{7.3}$$

in which $d(N_i, y)$ is the Euclidean distance between sensor N_i and point y, α is the energy emitted by events occurring at point y, and β is the decaying factor of the sensing signal. For radio signal sensing, β typically ranges from 2 to 5.

To describe the event sensibility in a region with cooperation of deployed sensors, it is convenient to introduce the concept of the sensing field [83], a corresponding concept to that of an electric field with a distribution of charges.

Definition 4 Suppose we have a "background" distribution of $|\Omega|$ sensors in a deployment region, and measure the sensibility on an event occurring at a point y. The sensing field associated with this sensor distribution is defined through the relation[1]

$$S_c(y) = \sum_{i \,:\, s(N_i,\, y) \,\geq\, \epsilon_n} s(N_i, y), \tag{7.4}$$

in which ϵ_n is the signal threshold due to measurement noise in the sensing environment, and $S_c(y)$ is called the *Collective-Sensor Field Sensibility* (CSFS) at point y.

With a required sensibility threshold ϵ_s, if $S_c(y) > \epsilon_s$, we say that point y is covered. If for every point in a deployment region, its CSFS is greater than ϵ_s, we say that the deployment region is covered.

[1]This relation is only applied to sensing signals that are additive, such as that of omni-directional acoustic sensors. If not, this equation should be redefined. However, the key point is that sensors are cooperated to detect an event.

If we determine a point's coverage based on the CSFS, we need a sink working as a data fusion center, who collects the signal intensities perceived by all sensors. Therefore, directly utilizing the CSFS to evaluate whether a point is covered will produce a heavy network load in multi-hop sensor networks and pose a single point of failure. Applying the fact that radio transmissions are non-directional in wireless sensor networks, we treat each sensor as a sensing fusion center[2] and introduce the following concept.

Definition 5 The *Neighboring-Sensor Field Sensibility* (NSFS) of sensor N_i at point y is defined by

$$S_n^i(y) = s(N_i, y) + \sum_{j \,:\, N_j \,\in\, N(i) \,\wedge\, s(N_j,\, y) \,\geq\, \epsilon_n} s(N_j, y). \tag{7.5}$$

When an event occurs at a certain point, the sensors that receive the signal will broadcast their perceived event sensibility. Each sensor calculates its NSFS after receiving the broadcast messages from its neighbors. If there is at least one sensor whose integrated field sensibility is greater than the sensibility threshold ϵ_s, then we say this point is covered. Thus we transform the originally global coverage decision problem into a local decision problem, and avoid producing a single point of failure.

Definition 6 A point y in a deployment region Φ is covered by a sensor set Ω when there is at least one sensor whose NSFS at point y is greater than or equal to a predefined threshold ϵ_s. Formally, point y is covered if $\exists N_i \in \Omega$, $S_n^i(y) > \epsilon_s$. Thus, a deployment region Φ is covered if all measured points in this region are covered.

The NSFS of sensor N_i considers only the sensibility provided by its one-hop neighbors and itself. Obviously, it is not greater than the CSFS, i.e.,

[2]This assumption simplifies the implementation of clustering algorithms [13, 55] as from the viewpoint of coverage each working sensor has the eligibility to be a cluster-head.

$S_n^i(y) \leq S_c(y)$. Note that the coverage problem is intrinsically global in the sense that lack of knowledge of the location of any single sensor implies that the problem may not be solved correctly [91]. But allowing for the possibility of missing some information in the coverage decision provides some redundancy, which is beneficial in building dependable sensor networks. We will identify this property later.

7.1.3 Relations between the BSM and the CSM

With (7.3) and a desired sensibility threshold ϵ_s, we can define an *ensured-sensibility radius* sr_i^e for sensor N_i as

$$sr_i^e = \left(\frac{\alpha}{\epsilon_s}\right)^{\frac{1}{\beta}}. \tag{7.6}$$

If an event occurs within the ensured-sensibility radius of a sensor, it should be detected by this sensor; therefore, the ensured-sensibility radius sr_i^e is equivalent to the sensing radius sr_i defined in the BSM. As a result, the CSM will be reduced to the BSM if we consider only the event detectability in sr_i^e.

Furthermore, we define a *collaborative-sensibility radius* sr_i^c on the basis of (7.3) and a signal threshold ϵ_n as

$$sr_i^c = \left(\frac{\alpha}{\epsilon_n}\right)^{\frac{1}{\beta}}. \tag{7.7}$$

For all events occurring in the collaborative-sensibility radius of a sensor, this sensor will contribute its sensibility to this event; however, a sensor does not perceive any event occurrence out of its collaborative-sensibility radius due to Gaussian noise.

Usually, ϵ_s is much greater than ϵ_n. Thus, sr_i^e is less than sr_i^c. With the BSM, each sensor can assure whether an event will be detected by itself. In addition to the event detectability provided by the BSM, the CSM may successfully detect an event that is undetected in the BSM by a cooperative process.

7.1.4 Voronoi Diagram

Voronoi diagram [11, 101], composed of a set of sensors, partitions a constrained two-dimensional sensor deployment region into a set of convex polygons such that all points inside a polygon are closest to only one particular sensor. These polygons are called Voronoi cells with finite areas as sensors are deployed in a constrained region. The boundary segment of a Voronoi cell is called the Voronoi edge shared by two sensors, and the intersection point of two Voronoi edges is called the Voronoi vertex shared by three or more sensors. The shared Voronoi edge of two sensors is on the perpendicular bisector line of a segment connecting these two sensors.

7.2 Sleeping Candidate Conditions

A sleeping sensor means its sensing devices and communication transceivers are turned off to save energy and to reduce packet transmission collision, i.e., it does not monitor its environment and does not send or receive messages. As a result, the network topology will be changed and the field sensibility of some regions will be reduced. We define the initial *covered area* to be the percentage of the deployment region that satisfies the coverage requirement with randomly scattered sensors. If we can ensure that there is no reduction on covered area after a sensor goes to sleep and the constructed network backbone is connected, this sensor is called a *sleeping candidate*. Otherwise, this sensor should keep working to provide its sensibility. With different characteristics of different sensing models, their corresponding sleeping candidate conditions should be different. Let us discuss the BSM case first.

7.2.1 Sleeping Candidate Condition for the BSM with Arc-Coverage

In the BSM, a sensor N_i only provides its sensibility in its sensing radius sr_i and it does not provide any sensibility out of this range. Therefore, to evaluate whether a sensor is a sleeping candidate with the BSM, we can only examine its sensing disk how it is covered by the sensing disks of its working neighbors. The approaches employed by previous work can be classified into three categories: by calculating the coverage of the sensing disk directly [126, 136], by evaluating the coverage of the sensing perimeter indirectly [58], or by investigating the coverage of the intersection points of the sensing perimeters indirectly [133, 143]. Directly calculating the coverage of a sensing disk may time-consuming as all measuring points should be evaluated [136] or may underestimate the coverage provided by neighboring sensors [126]. Indirectly investigating the coverage of the intersection points of the sensing perimeters overcomes those disadvantages; however, it cannot estimate the reduced area when allowing area coverage loss. Therefore, in this subsection, we develop a sleeping candidate condition based on arc-coverage [29]. Its coverage requirement is that when a sensor goes to sleep, all measuring points in its sensing disk will be still covered by at least k sensors, i.e., the k-coverage requirement. Some additional definitions are introduced to help us develop the k-coverage sleeping candidate condition.

Definition 7 Suppose sensors N_i and N_j are one-hop communication neighbors, and their sensing perimeters intersect at points p_{ij}^1 and p_{ij}^2 which are arranged in the clockwise order with respect to N_i. As illustrated in Fig. 7.1, the part of the sensing disk of N_i that is also covered by N_j is the shaded

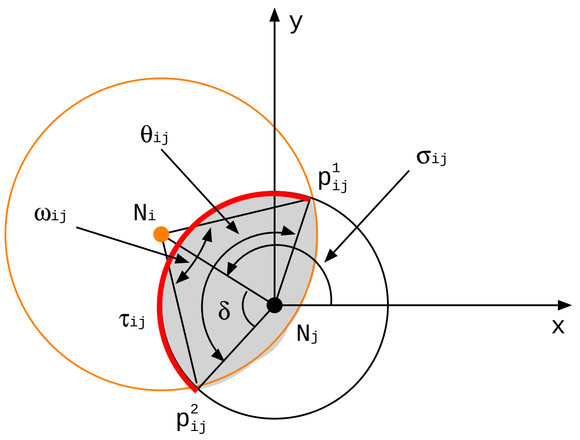


Figure 7.1: Sponsored sensing region, arc and angle and covered sensing angle

region, which is called the *Sponsored Sensing Region* (SSR) by N_j to N_i and equals $\Psi_i \cap \Psi_j$. The arc $\widehat{p_{ij}^1 p_{ij}^2}$ on the sensing perimeter of N_j is defined as the *Sponsored Sensing Arc* (SSA), denoted as τ_{ij} (shown with a heavy arc in Fig. 7.1), and its corresponding central angle is called the *Sponsored Sensing Angle* (SSG), denoted as θ_{ij}. Note that the points on τ_{ij} are not covered by N_j, according to Equation (7.2). The direction of N_i referred to N_j is denoted as σ_{ij}; therefore, $\theta_{ij} = (\sigma_{ij} - \delta, \sigma_{ij} + \delta)$, in which δ is a half of the central angle θ_{ij}. Note that θ_{ij} expresses the relative position of τ_{ij} on the perimeter of Ψ_j. In addition, we let $\omega_{ij} = \angle p_{ij}^1 N_i p_{ij}^2$, which is called the *Covered Sensing Angle* (CSG). Actually, $\omega_{ij} = \theta_{ji}$; however, they denote different physical meanings. θ_{ji} is a measure of SSA τ_{ji}, whereas ω_{ij} implies which part of the perimeter of Ψ_i is covered by N_j.

From geometry calculations, we know

$$\sigma_{ij} = \begin{cases} \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) & : \ if \ \ x_j - x_i < 0 \ \ \wedge \ \ y_j - y_i < 0 \\ \pi + \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) & : \ if \ \ x_j - x_i > 0 \\ 2\pi + \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right) & : \ if \ \ x_j - x_i < 0 \ \ \wedge \ \ y_j - y_i > 0 \end{cases} \tag{7.8}$$

and

$$\delta = \arccos\left(\frac{sr_j^2 + d^2(N_i, N_j) - sr_i^2}{2 \cdot sr_j \cdot d(N_i, N_j)}\right). \tag{7.9}$$

The above definitions apply to the case in which the sensing perimeters of two neighboring sensors have intersection points. If the sensing disks of two sensors are overlapped, each of them is denoted as a *coverage sponsor* to another. There are three special cases which all lack intersection points between two sensor's sensing perimeters, as shown in Figure 7.2. The first

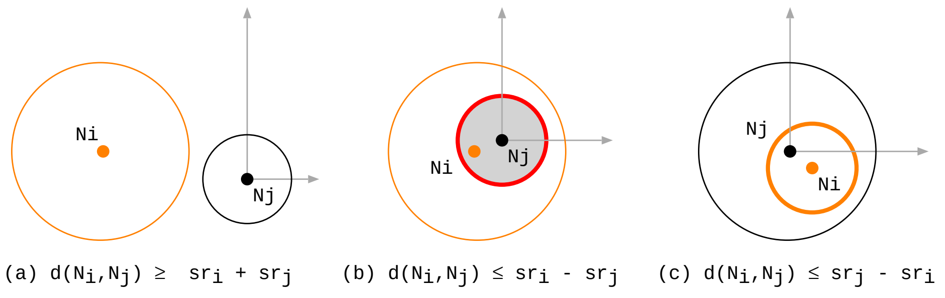


Figure 7.2: Special cases of sponsored sensing region and arc

one is the case when $d(N_i, N_j) \geq sr_i + sr_j$. In this case, there is no overlap between the sensing disks of two neighboring sensors and N_j does not provide any coverage to N_i. For the second case when $d(N_i, N_j) \leq sr_i - sr_j$, Ψ_j is completely covered by N_i; therefore, the SSR is all the sensing disk of N_j, the SSA τ_{ij} is the perimeter of N_j, and SSG $\theta_{ij} = 2\pi$. However, CSG ω_{ij} is not defined. The last case is when Ψ_i is completely contained in Ψ_j, which happens whenever $d(N_i, N_j) \leq sr_j - sr_i$ holds. In this case, the SSR, the SSA, and the SSG are not defined; however, the CSG is defined as $\omega_{ij} = 2\pi$. An additional concept is introduced to deal with this case.

Definition 8 The number of working neighbors whose sensing disks completely contain the sensing disk of sensor N_i is called the *Degree of Complete Coverage* (DCC) sponsored by neighboring sensors, denoted as ζ_i. These neighboring sensors are called *Complete-Coverage Sponsors* (CCSs) of N_i, denoted as $CCS(i)$, and other working neighbors are called non-CCSs of N_i. Therefore, the set of CCSs of N_i is denoted as $CCS(i) = \{N_j \in N(i) | \ d(N_i, N_j) \leq sr_j - sr_i\}$.

Definition 9 With two sensor nodes N_i and N_j, on SSA τ_{ij} we find a point y that is covered by the minimum number of sensors. Then the number of N_i's non-CCSs covering the point y is defined as the *Minimum Partial Arc-Coverage* (MPAC) sponsored by node N_j to node N_i, denoted as ξ_{ij}.

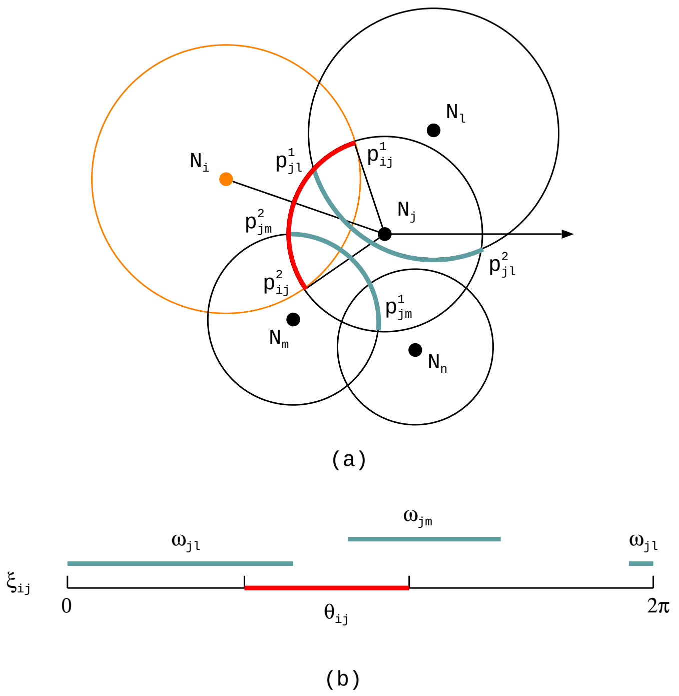


Figure 7.3: Derivation of the MPAC ξ_{ij} sponsored by sensor N_j to sensor N_i

Fig. 7.3 illustrates a calculation of MPAC ξ_{ij}. The calculation steps are as follows [58]:

1. Draw a line segment representing $[0, 2\pi]$;

2. Calculate SSG θ_{ij}, and mark this angle accordingly on the line;

3. For each non-CCS of N_i, calculate its CSG related to N_j and lay out the derived CSG proportionally on the line;

4. Scan the line segment θ_{ij} and write down the minimum number that the points on this line segment are covered by the SSG and CSGs.

The derived minimum number is MPAC ξ_{ij}. For the case illustrated in Fig. 7.3, we get $\xi_{ij} = 1$. The critical part of SSA τ_{ij} is $\widehat{p_{jl}^1 p_{jm}^2}$ as this arc is covered by the minimum number of sensors and determines the MPAC. From the definition and calculation steps, we know $\forall\ N_i \in \Omega$, $\xi_{ij} \geq 1$, because all points on SSA τ_{ij} are at least covered by N_i itself.

MPAC and DCC Based k-Coverage Sleeping Candidate Condition

The statement "a region is k-covered" means every point inside this region is covered by at least k sensors. If the coverage degree of the sensing disk of a sensor is at least $(k + 1)$, then this sensor could fall asleep to save energy without compromising the network's overall k-coverage requirement and without reducing area coverage. For sensor N_i, we define its neighboring index set $I_i = \{m|N_m \in N(i)\}$, $m = 1, 2, 3, \ldots$. Then the general k-coverage sleeping candidate condition can be expressed as $\Psi_i \subseteq \bigcup_{j \in I_i^n} \Psi_j$, in which $n = 1, \ldots, k$, $I_i^n \subseteq I_i$, and $\forall\ 1 \leq p, q \leq k$, $I_i^p \cap I_i^q = \emptyset$ when $p \neq q$. This general sleeping candidate condition is theoretically accurate under the constraint of only one-hop neighbors but cannot be directly employed to select sleeping candidates. We utilize the proposed MPAC and DCC to decide which sensors are sleeping candidates while preserving k-coverage.

Theorem 4 A sensor N_i is a sleeping candidate while preserving k-coverage under the constraint of one-hop neighbors, iff its DCC is greater than or equal to k, or it has at least one non-CCS neighbor and for all its non-CCS neighbors

$N_j \in N(i) - CCS(i)$, the MPAC sponsored by N_j to N_i is greater than k minus N_i's DCC. Formally, N_i is sleeping-eligible iff $\zeta_i \geq k$, or $N(i) - CCS(i) \neq \emptyset$ and $\forall\ N_j \in N(i) - CCS(i)$, $\xi_{ij} > k - \zeta_i$.

Proof: In order for sensor N_i to be a sleeping candidate, all points in its sensing disk Ψ_i should be covered by at least $(k+1)$ nodes including N_i. As its DCC is ζ_i, all points in Ψ_i are already covered by at least $(\zeta_i + 1)$ nodes. If $\zeta_i \geq k$, then obviously N_i is sleeping-eligible. If $\zeta_i < k$, we should ensure that its non-empty non-CCS neighbors providing residual $(k - \zeta_i)$-coverage. This is equivalent to proving that if N_i is not sleeping-eligible and $N(i) - CCS(i) \neq \emptyset$, there exists a node $N_j \in N(i) - CCS(i)$, $\xi_{ij} \leq k - \zeta_i$. In the sensing disk of a sensor, the minimally covered points are always on SSAs when $N(i) - CCS(i) \neq \emptyset$. As N_i is not sleeping-eligible, some points on the SSAs are covered by less than $(k - \zeta_i)$ non-CCS neighbors. Without loss of generality, we assume that one of these SSAs is sponsored by N_j. Then from Definition 9, we have $\xi_{ij} \leq k - \zeta_i$.

For the "only if" part, we should prove if N_i is sleeping-eligible, then $\zeta_i \geq k$, or $N(i) - CCS(i) \neq \emptyset$ and $\xi_{ij} > k - \zeta_i$ for all sensors $N_j \in N(i) - CCS(i)$. This is equivalent to proving that when $\zeta_i < k$, and there exists a sensor N_j whose MPAC to N_i is less than or equal to $(k - \zeta_i)$, i.e., $\xi_{ij} \leq k - \zeta_i$, sensor N_i is not sleeping-eligible. If $\zeta_i < k$ and $\xi_{ij} < k - \zeta_i$, obviously N_i is not sleeping-eligible. If $\zeta_i < k$ and $\xi_{ij} = k - \zeta_i$, some points on SSA τ_{ij} are covered by only k sensors. Note that these points are in Ψ_i; however, they are not in Ψ_j. If N_i goes to sleep, these points will be covered by at most $(k-1)$ sensors; then k-coverage will not be preserved. Therefore, when $\zeta_i < k$ and $\xi_{ij} = k - \zeta_i$, N_i is not sleeping-eligible. ■

Theorem 4 can be extended to deal with irregular and/or non-uniform sensing disks as long as the sensing disk of each sensor can be precisely defined.

Extended Sleeping Candidate Condition

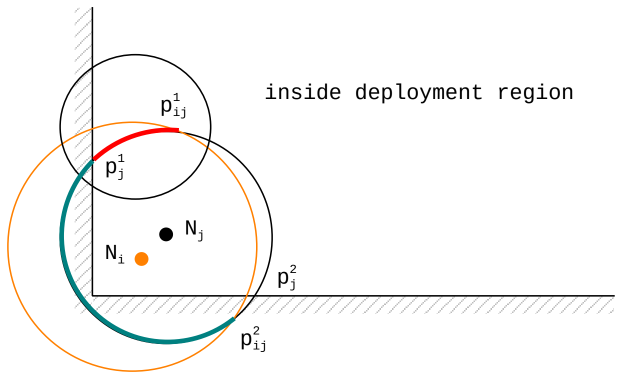


Figure 7.4: Sponsored sensing arcs in a constrained deployment region

When we apply the MPAC and DCC based sleeping candidate condition to schedule sensors' sleeping in a restricted two-dimensional deployment region, more sensors will be identified as sleeping candidates if we take the boundary case into consideration, illustrated in Figure 7.4. We denote the original sleeping candidate condition as *Mpac*, and the sleeping candidate condition which takes the boundary case into consideration as *MpacB*. To determine whether sensor N_i is a sleeping candidate, we now need only consider its sensing disk inside the monitoring area. Therefore, some parts of the original SSA τ_{ij} are removed and we must test only arcs $\widehat{p_{ij}^1 p_j^1}$. As a result, N_i is not sleeping-eligible for 1-coverage with *Mpac*; however, it is a sleeping candidate with *MpacB*. With this extension, sleeping-eligible sensors will be identified even in irregular areas or areas with obstructions.

When each sensor evaluates its sleeping eligibility with MPAC, it identifies which parts of SSAs on its sensing perimeters are critical arcs. A critical arc is the part of an SSA and is covered by the minimum number of sensors. These critical arcs will form some subregions which do not satisfy the required degree of coverage when the sensor goes to sleep; therefore, we call these subregions

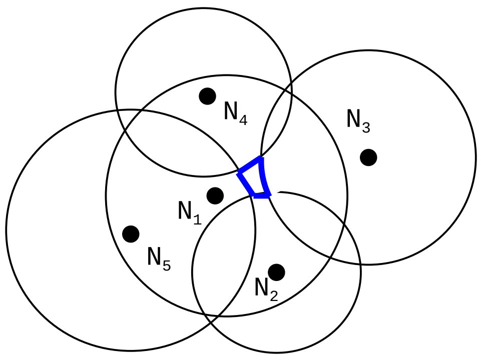


Figure 7.5: Critical arcs in evaluating N_1's sleeping eligibility

as *critical regions*. If we could estimate the size a critical region to be smaller than a predefined threshold, we may opt to omit this region, as the coverage requirement is still largely satisfied. This may lead to the sensor becoming sleeping-eligible. With this critical region extension added to the model, more sensors can sleep at any given time; however, the cost we must pay is the loss of area coverage. The thick arcs in Figure 7.5 are critical arcs, and they form a critical region for 1-coverage. This region is small; therefore, even sensor N_1 goes to sleep, the loss of area is negligible. Let *MpacBCa* denote the critical region extension to *MpacB*. We utilize the length of critical arc to estimate the size of its formed critical region. If the length of a critical arc is less than a threshold, we treat this critical arc as a normal arc; when all critical arcs bounding a critical region are treated as normal arcs, the region is no longer considered critical. A special case is when the threshold is 0, *MpacBCa* is reduced to *MpacB* because no critical arcs are treated as normal.

7.2.2 Sleeping Candidate Condition for the BSM with Voronoi Diagram

All previous work [58, 126, 133, 136, 143] and the proposed sleeping candidate condition *Mpac* in Subsection 7.2.1 are based on the geometry calculation of the sensing disk. Inspired by the concept of coverage boundary introduced by Carbunar *et al.* in [17], in the subsection we develop a sleeping candidate condition with the property of Voronoi diagram. It evaluates the coverage of Voronoi vertices instead of the coverage of the sensing perimeters and the coverage of their intersection points [31].

Definition 10 A sensor N_i is said to be on the boundary of coverage if there exists a point y on its sensing perimeter such that y is not covered by its one-hop working neighbors $N(i)$.

This definition considers only a sensor's one-hop and working neighbors but not all other sensors, which extends the corresponding concept in [17]. Therefore, it is more suitable to develop distributed and localized sleeping configuration algorithms, which is presented Section 7.3.

A theorem to evaluate whether a sensor is on the boundary of coverage or not is also provided in [17]:

Theorem 5 A sensor N_i is on the boundary of coverage if and only if its Voronoi cell is not completely covered by its sensing disk.

Figure 7.6[3] gives an example of coverage boundary. For sensor N_1, all other sensors, N_i, $i = 2, \ldots, 8$, are its one-hop working neighbors. The small squares represent the locations of sensors, the circles are the sensing perimeters, and the line segments are the Voronoi diagram constructed by these sensors. The outer rectangle is the constrained deployment region, thus each Voronoi cell is finite. As the sensing disk of N_1 does not cover its Voronoi cell, N_1 is on the coverage boundary. Obviously, if a sensor is on the coverage boundary, it is not sleeping-eligible as some parts of its sensing disk are only covered by

[3]Figures with Voronoi diagrams are generated on the basis of the VoroGlide provided by Icking *et al.* at http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide

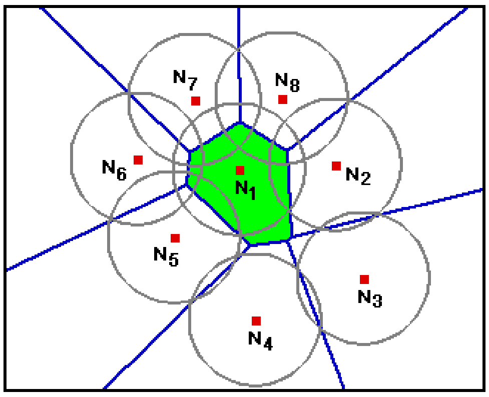


Figure 7.6: Example of coverage boundary: N_1.

itself and its sleeping will reduce the covered area. Thus, N_1 is not a sleeping candidate.

To facilitate the evaluation process, we provide a corollary that can be easily derived from Theorem 5:

Corollary 2 A sensor N_i is on the boundary of coverage if and only if there exists one of its Voronoi vertices that is not in its sensing disk.

In principle, a sensor is a sleeping candidate if its sleeping does not reduce the covered area when it works. Therefore, if a sensor is on the coverage boundary, it cannot be sleeping-eligible. However, even if it is not on the coverage boundary, it may also not be sleeping-eligible. An example is that a sensor's sensing perimeter is all covered, but some inner parts of its sensing disk are only covered by itself. We need to provide a necessary and sufficient condition to evaluate a sensor's sleeping eligibility, which is provided as the following theorem.

Theorem 6 A sensor N_i is a sleeping candidate if and only if (1) it is not on the coverage boundary; and (2) when constructing another Voronoi diagram without N_i, all the Voronoi vertices of its one-hop working neighbors in N_i's sensing disk are still covered.

Proof: If sensor N_i is not on the coverage boundary, according to Theorem 5, its Voronoi cell is completely covered by its sensing disk, i.e., its Voronoi cell is in its sensing disk. As a result, in the regenerated Voronoi diagram without N_i, all vertices outside N_i's sensing disk are not changed, and no new vertices are generated outside N_i's sensing disk. This is because when removing a sensor, all the Voronoi cells of its neighbors will be enlarged. If all these Voronoi vertices are still covered, its neighbors should not become sensors on the coverage boundary due to N_i's sleeping. Thus no covered area is reduced. Therefore, N_i is a sleeping candidate.

For the "only if" part, if N_i is a sleeping candidate, (1) it is not on the coverage boundary, and (2) in the regenerated Voronoi diagram without N_i, all Voronoi vertices of its neighbors in N_i's sensing disk are still covered. First, let us assume that N_i is a sleeping candidate and it is on the coverage boundary. If a sensor is on the coverage boundary, parts of its sensing perimeter are only covered by itself. As a result, when N_i goes to sleep, these parts of its sensing perimeter are not covered by its neighbors, thus reducing the covered area. This leads to a contradiction. Second, let us assume that N_i is a sleeping candidate and one of those regenerated Voronoi vertices in N_i's sensing disk is not covered. As the uncovered Voronoi vertex is in N_i's sensing disk, this vertex is covered only by N_i if N_i is working. Therefore, N_i's sleeping results in this vertex being uncovered, i.e., N_i's sleeping reduces the covered area. This leads to a contradiction again. ■

Figure 7.7 shows a sleeping-eligible sensor N_1. Figure 7.7(a) shows when N_1 is working, it is not on the coverage boundary, and its Voronoi cell is completely contained in its sensing disk. Figure 7.7(b) shows the regenerated Voronoi cell when N_1 goes to sleep. All the Voronoi vertices in its sensing disk are still covered by other sensors; therefore, N_1 is a sleeping candidate.

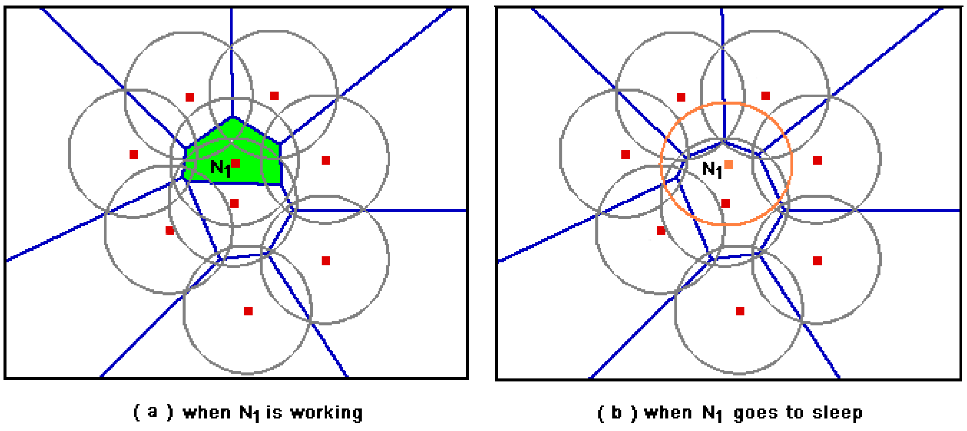


Figure 7.7: Example of sleeping-eligible sensor: N_1.

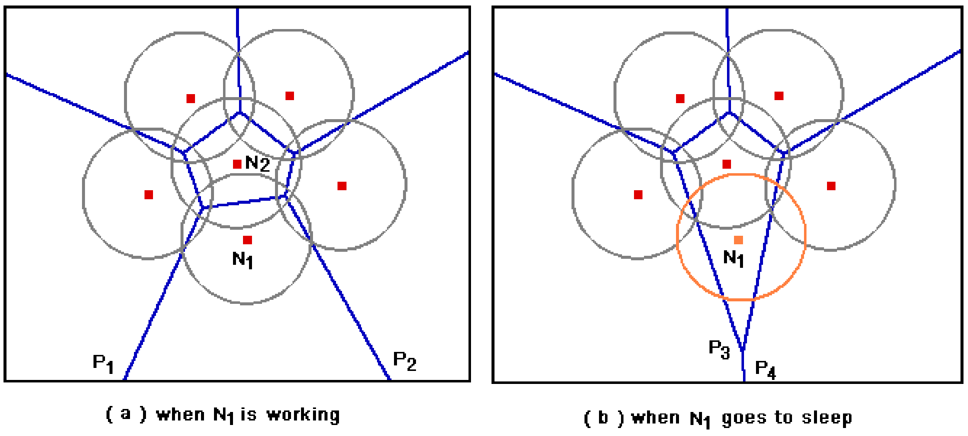


Figure 7.8: Example of necessary condition: N_1.

The first condition that a sensor is not on the coverage boundary should be included as a necessary condition for the sensor to be sleeping-eligible, as illustrated in Figure 7.8. If we evaluate N_1's sleeping eligibility only based on the second condition, N_1 is a sleeping candidate as no Voronoi vertex in its sensing disk is not covered, as shown in Figure 7.8(b). However, we know that N_1 is not sleeping-eligible from Figure 7.8(a) as it is on the coverage boundary. The reason is that the first condition ensures that no Voronoi vertex outside a sensor's sensing disk would be changed due to the sensor's presence or absence.

As a result, we can only evaluate the coverage of the Voronoi vertices in its sensing disk. As N_1 does not satisfy the first condition, it is easily observed from Figure 7.8 that some old vertices, P_1 and P_2, outside the sensing disk of N_1 disappear, and some new vertices, P_3 and P_4, are generated. Figure 7.8(a) also shows that sensor N_2 is not on the coverage boundary but it is not sleeping-eligible.

When a sensor utilizes all its one-hop working neighbors to calculate its sleeping eligibility, we get a 1-coverage sensor sleeping configuration. To obtain k-coverage, $k \geq 2$, we may divide its one-hop working neighbors into k mutually disjunct subsets. If a sensor satisfies the sleeping candidate condition (Theorem 6) with each subset of its neighbors, then we can say the sensor is a sleeping candidate for k-coverage.

The locally constructed Voronoi diagram with only one-hop neighbors may be an approximation to the Voronoi diagram generated by a centralized computation with the information of all deployed sensors. For simplicity, we denote the maximum of the sensing radii of all deployed sensors as sr. When the communication radius cr is at least twice of the sensing radius sr, a sensor can find all other sensors which provide their sensibility to this sensor's sensing disk with only one-hop communication [133, 143]. Otherwise, the sensor can learn those sensors within $\lceil 2 \cdot sr/cr \rceil$ hops. Our approach ignores the sensibilities contributed by one-hop out-of-reach sensors and so underestimates the coverage of a sensor's sensing disk by using partial sensor deployment information only. However, this underestimation is beneficial to building dependable wireless sensor networks, and reduces the overload on the network due to sleeping configuration. Therefore, the major computation incurred in this sleeping candidate condition is to calculate the Voronoi vertices generated by a sensor's one-hop neighbors. It can be solved in $O(n \log n)$, where n is the number of the sensor's one-hop neighbors. As this number is usually not very large, the resulted computational cost is also acceptable.

7.2.3 Sleeping Candidate Condition for the CSM

In the BSM, a sensor N_i confines its sensibility contribution within its sensing radius sr_i; however, in the CSM, a sensor extends its contribution to its collaborative-sensibility radius sr_i^c. For a measuring point that is outside a sensor's sensing radius while in its collaborative-sensibility radius, the sensor cannot determine whether this point is covered by itself only; it needs to collect the sensibility contribution information from its neighbors.

When introducing the concept of the NSFS, we have stated that if an event occurs at a certain measuring point, the sensors that receive the signal will broadcast their perceived event sensibility. Therefore, a sensor contributes its sensibility to and only to its one-hop neighbors. No other sensors will receive its broadcasting message. After it falls asleep, it becomes invisible to all its neighbors, thus the Voronoi diagram will be changed and the Voronoi cells of its neighbors will be enlarged to cover the sleeping sensor's original Voronoi cell. According to the definition of a Voronoi diagram, if an event occurs in a Voronoi cell, its corresponding sensor will receive the strongest signal. Thus its NSFS will most likely satisfy the coverage requirement. Consequently, we need only assess the enlarged Voronoi cells of a sensor's one-hop neighbors when evaluating its sleeping eligibility. If all these Voronoi cells are still covered with a sensibility threshold ϵ_s, then we can say that it is safe to allow a sensor to sleep.

For a sensor to exactly learn the Voronoi cells of its one-hop neighbors, the first approach is for it to know the locations of all deployed sensors that share the Voronoi edges with its one-hop neighbors. Those sensors, however, may be far away and cannot be reached within a few hops. The other approach is to provide a centralized service that calculates every sensor's Voronoi cell and dispatches this information to all sensors. Both of these scenarios are undesirable. We need to solve this problem based on local information only; however, computing the neighbors' Voronoi cells exactly with local information only may be impossible. Moreover, some redundancy should be kept to tolerate sensor failures and energy depletion. As a trade-off, here we employ the two-hop neighbors' information to produce a sensor's local view of the Voronoi diagram composed by its neighbors. In this way, we pessimistically enlarge the Voronoi cells of some neighbors and augment their responsibilities. But all these Voronoi cells are confined by a sensor's collaborative-sensibility radius r_c as a sensor does not contribute its sensibility out of this radius.

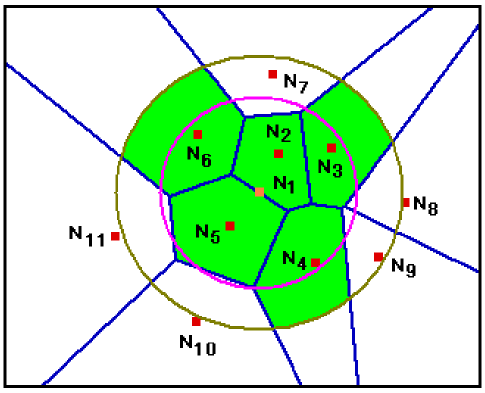


Figure 7.9: Scan region for sensor N_1.

Figure 7.9 illustrates a deployment of sensors. All sensors are N_1's one- and two-hop neighbors. A corresponding Voronoi diagram from the viewpoint of sensor N_1 is also given when it goes to sleep. The inner circle represents N_1's one-hop communication radius, and the outer circle denotes its collaborative-sensibility radius. Sensors N_i, $i = 2, \ldots, 6$, are N_1's one-hop neighbors and sensors N_i, $i = 7, \ldots, 11$, are N_1's two-hop neighbors. The shaded areas are the confined Voronoi cells that must be assessed to evaluate N_1's sleeping eligibility. The N_1's two-hop neighbors do not consider its sensibility; therefore, even parts of their Voronoi cells are inside N_1's collaborative-sensibility radius, whose areas do not need to be evaluated. (7.3) shows the nonlinearity in the sensor sensibility with respect to the signal strength, which prohibits derivation of a simple geometry computation to determine the coverage of the Voronoi cells of a sensor's one-hop neighbors. Therefore, we cover these cells with a virtual square grid and calculate the sensibility of each sampling point at the center of each grid [134, 136, 145]. The granularity of this grid depends on the trade-off between computation efficiency and coverage preservation.

7.2.4 Location Error

In all the aforementioned sleeping candidate conditions, each sensor knows its accurate location. However, this is not realistic [114, 119]. Here we assume that a sensor's obtained location is uniformly distributed in a circle located at its accurate position with radius ϵ_d. We call the ratio of the maximum location deviation ϵ_d to a sensor's sensing radius the *normalized deviation of location* ϵ, and the ratio of the distance between a point and a sensor to the sensor's sensing radius the *normalized distance d*. Without location error, when the normalized distance is less than 1, the point is deterministically covered by the sensor. Nevertheless, with location error, all points in $(1 + \epsilon)$ will be covered by the sensor with uncertainty.

Figure 7.10 shows the coverage cases with different normalized distances, and Figure 7.11 depicts the corresponding probability of coverage. In the former figure, a small solid circle denotes a sensor's obtained location, a dashed circle represents its normalized deviation of location ϵ, and a cross expresses the evaluated point. All points outside the outermost circle cannot be covered.

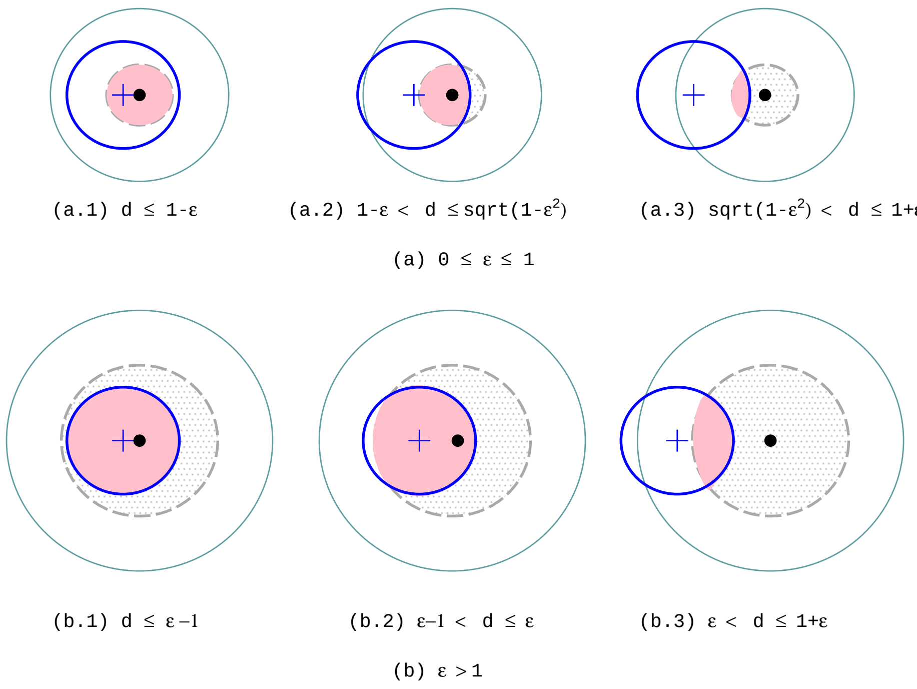


Figure 7.10: The coverage relationship between a point and a sensor with location error.

If the sensor is located in the shaded region, the evaluated point can be covered. Therefore, the probability of coverage is the ratio of the area of the shaded region to the area of location deviation $\pi\epsilon^2$. Figure 7.10(a.1) shows when $0 \leq \epsilon \leq 1$, all points in distance $(1 - \epsilon)$ are still covered with probability 1; however, when $\epsilon > 1$, the maximum probability of coverage is $1/\epsilon^2$, as shown in Figure 7.10(b.1).

With location error, after sleeping configuration we cannot ensure there is no loss of area coverage for the BSM; however, we can ensure that the uncovered area is less than a predefined threshold by reducing the sensing radii of deployed sensors during evaluation of sleeping eligibility. Given a normalized deviation of location ϵ and a predefined coverage probability, a sensor gets its maximized sensing distance from Figure 7.11 and employs this sensing distance as its adjusted sensing radius.

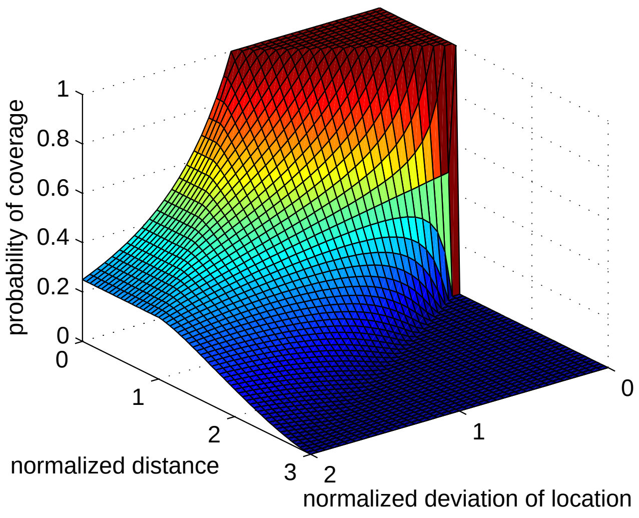


Figure 7.11: Probability of coverage with location error.

For the CSM, we extend the NSFS defined in (7.5) with location error by introducing the probability of coverage $p(N_i, y)$ as follows:

$$S_n^i(y) = p(N_i, y)s(N_i, y) + \sum_{j \,:\, N_j \,\in\, N(i) \,\wedge\, s(N_j,\, y) \,\geq\, \epsilon_s} p(N_j, y)s(N_j, y), \quad (7.10)$$

in which $p(N_i, y)$ denotes the probability of point y covered by sensor N_i.

7.2.5 Network Connectivity

When detecting an event, sensors report this event to data sinks. Therefore, the network should be connected to successfully perform its sensing and monitoring task. Considering only the sensibility issue when evaluating a sensor's sleeping eligibility may produce disconnected subnetworks, and as a result, even though an event is successfully detected by sensors, this information may not be delivered to the data sinks. To construct an effective sensor network, we must take the communication connectivity into consideration.

For the BSM, a theorem has been proved [133, 143]: If the communication radius cr is at least twice of the maximal sensing radius sr, preserving area coverage implies maintaining network connected. However, it is only valid for accurate location information. Therefore, we evaluate whether a sensor's one-hop working neighbors will remain connected through each other when this sensor is removed. For the CSM, we relax this one-hop connectivity checking to two-hop neighbors as a sensor has collected all its two-hop neighbors. Then, we say that sensor N_i is sleeping-eligible if all the Voronoi cells, that are inside its collaborative-sensibility radius, of its one-hop working neighbors are covered, and its one-hop working neighbors will remain connected through each other or through its two-hop working neighbors when N_i is removed. As no location information is engaged in this evaluation, it is valid under location error. Obviously, if all N_i's one-hop neighbors are connected, its two-hop neighbors also are connected because these two-hop neighbors are connected with at least one of its one-hop neighbors.

The connectivity check through a sensor's one- or two-hop neighbors is heuristic, but simulation results show it performs well in maintaining a connected network.

7.3 Sensibility-Based Sleeping Configuration Protocol (SSCP)

Until now, we have introduced three sleeping candidate conditions: two for the BSM and one for the CSM. We can apply these conditions for a sensor to evaluate whether it is sleeping-eligible or not. However, if we simply schedule all sleeping-eligible sensors to turn off their sensing devices, a covered area reduction may be produced when two compensative sensors go to sleep together. Two sensors are defined as *compensative sensors* when, if either of them goes into sleeping status, no covered area will be reduced; however, if both of them are sleeping, covered area must be reduced. Therefore, sensors should be carefully coordinated to negotiate which sleeping-eligible sensors go to sleep. We develop here two decentralized and localized coordination protocols: round-based and adaptive sleeping, to schedule sensors' on and off time properly in order to conserve energy and reduce packet collision whilst preserving area coverage. Therefore, with a set of deployed sensors Ω, the objective of sleeping configuration is to find a subset of working sensors with which the covered area of a deployment region is not reduced to be less than the covered area when all sensors are working while minimizing the number of working sensors and maintaining network connectivity. Then other sensors are scheduled to sleep.

7.3.1 Round-Based Sleeping Configuration Protocol

The round-based sleeping configuration protocol divides time into rounds. In each round, every live sensor is given a chance to be sleeping-eligible to balance energy depletion between sensors. It requires that sensors should be approximately synchronized [36, 88, 116]. As negotiating among two-hop neighbors is more complicated than that among one-hop neighbors, here we only address the sensibility-based sleeping configuration protocol (SSCP) for the CSM. The protocol for the BSM can be similarly derived.

In this protocol, six timers are employed. Timers $T_{hello-i}$ and $T_{hello-ii}$ are backoff timers for reducing the probability of packet collision. Timers T_{win-i} and T_{win-ii} are intended for collecting HELLO-I and HELLO-II messages, respectively. Timer T_{wait} is a window for compensative sensors to negotiate their cooperative status. Timer T_{round} divides the sensors' working time into cycles. At the start of each cycle, all live sensors compete to enter the sleeping mode. Because of this timer, a working sensor may get a chance to be sleeping-eligible, thus balancing the sensor's energy consumption and prolonging the lifetime of the network. Moreover, round-based reconfiguration engages a certain inherent immunity to sensor failures as it will attempt to recover covered area and network connectivity in the next round's sleeping configuration.

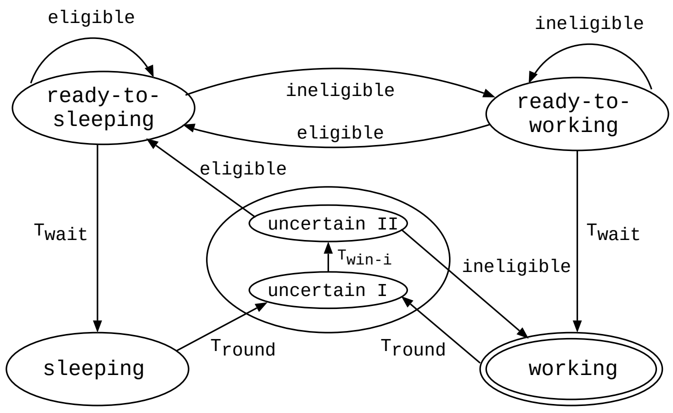


Figure 7.12: Sensor status transition in SSCP.

The corresponding sensor status transition diagram is shown in Figure 7.12, and the detail protocol steps will be given in the next paragraph. Initially all sensors are in the WORKING status. After starting configuration, all live sensors enter the UNCERTAIN-I status, in which they broadcast their own information and collect their one-hop neighbors. In the UNCERTAIN-II status, sensors broadcast their collected one-hop neighbors' information, and thus their two-hop neighbors' information will be obtained. The READY-TO-SLEEPING status and timer T_{wait} are intended to avoid compensative sensors entering the SLEEPING status at the same time. Timer T_{wait} terminates one round decision, and all live sensor are scheduled into either the SLEEPING or WORKING status. A sensor in READY-TO-WORKING may become sleeping-eligible if it receives a STATUS-I or STATUS-II message from a sensor who or whose one-hop neighbors will provide the residual field sensibility. After the first eligibility evaluation in each round, if a sensor is not a sleeping candidate, we set its status directly to WORKING. The reason is that even though it may receive subsequent STATUS-I and STATUS-II messages, it cannot be sleeping-eligible any more.

Initially, every sensor is in the WORKING status and sets a round timer T_{round}. When its round timer expires, the sensor sets its status to UNCERTAIN-I and enters the working-sleeping decision phase, which contains eight steps:

1. Setting a backoff timer $T_{hello-i}$, a window timer T_{win-i}, a wait timer T_{wait}, and the next round timer T_{round}; then starting to collect HELLO-I messages from neighbors and creating a neighbor list. The HELLO-I message conveys a sensor's ID and location.

2. After $T_{hello-i}$ times out, broadcasting a HELLO-I message to all one-hop neighbors.

3. After T_{win-i} expires, the HELLO-I messages sent by its one-hop neighbors have been collected, and it prepares to send out a list of its neighbors. Therefore, its status is changed to UNCERTAIN-II. It sets another backoff timer $T_{hello-ii}$ and a window time T_{win-ii}; then it starts to collect HELLO-II messages, which conveys the information of a sensor's one-hop neighbors.

4. After $T_{hello-ii}$ times out, broadcasting a HELLO-II message to all neighbors.

5. After T_{win-ii} expires, it learns all its one- and two-hop neighbors. Now it is ready to evaluate its sleeping eligibility according to the sleeping candidate conditions discussed above. If eligible, its status is set to READY-TO-SLEEPING and broadcasts a STATUS-I message; otherwise, its status is set to WORKING. The STATUS-I message contains a sensor's ID, current status and its one-hop neighbors information. The

purpose of this message is to alert other compensative sensors to reevaluate their sleeping eligibility as the sender has changed its status.

6. When receiving a STATUS-I message, updating the corresponding information in the neighbor list, then

    (a) If the sensor's own status is WORKING, other sensors' status does not affect its own status; however, as two-hop information is needed in our sleeping candidate condition, the sensor should forward the received STATUS-I message. Therefore, it constructs a STATUS-II message, which contains the same information as the STATUS-I message, and broadcasts it.

    (b) If the status is READY-TO-SLEEPING or READY-TO-WORKING, its status may be affected by other sensors; therefore, it reevaluates its sleeping eligibility and sets it status to READY-TO-SLEEPING or READY-TO-WORKING accordingly. If its status is not changed, it sends out a STATUS-II message; otherwise, a STATUS-I message will be dispatched, and no further STATUS-II message is required, as a STATUS-I message contains the same information as that of a STATUS-II message.

7. When receiving a STATUS-II message, also updating its neighbor list, then

    (a) If its own status is WORKING, nothing should be done.

    (b) If its status is READY-TO-SLEEPING or READY-TO-WORKING, it reevaluates its sleeping eligibility and changes its status accordingly. If its status is not changed, no further actions are required; otherwise, it sends out a STATUS-I message.

8. After T_{wait} times out, if its status is READY-TO-SLEEPING, setting the status to SLEEPING and entering the power saving mode; if the status is READY-TO-WORKING, setting the status to WORKING and keeping sensing.

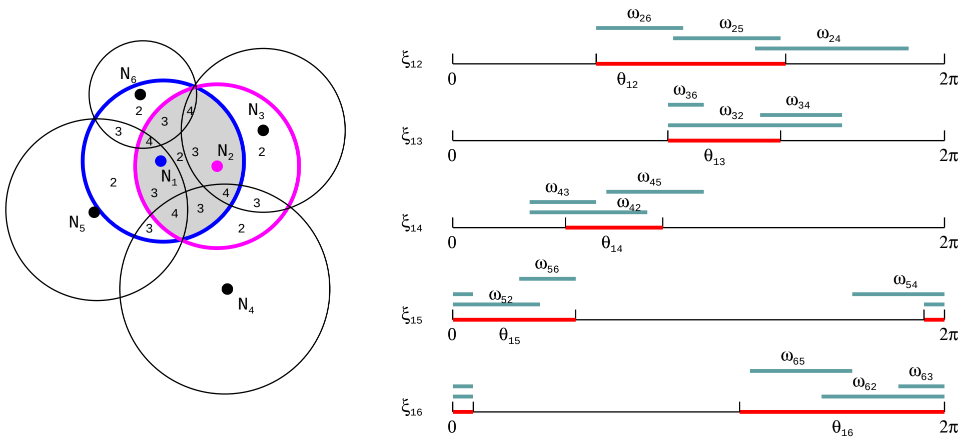


Figure 7.13: An example of sleeping eligibility evaluation for the BSM with arc-coverage.

Fig. 7.13 illustrates a sleeping eligibility evaluation for sensor N_1 using the sleeping candidate condition developed under the BSM with arc-coverage. For all neighbors of N_1, their MPACs are greater than or equal to 2. Therefore N_1 is 1-coverage sleeping-eligible. Additionally, we identify all critical regions for 2-coverage in Ψ_1, which are the areas whose coverage degree is 2 in this figure. N_2 is also a sleeping candidate; however, if both sensors are scheduled into sleeping, the part of the shaded area whose coverage degree is 2 will not be covered by any sensors, thus reducing the covered area. This is the reason why a sleeping-eligible sensor does not set its status to WORKING or SLEEPING directly after evaluating its sleeping eligibility. If N_2 learns that N_1 will be scheduled into sleeping from a STATUS message sent by N_1, it will be not sleeping-eligible anymore, thus preserving area coverage.

7.3.2 Adaptive Sleeping Configuration Protocol

A sensor may suffer failure or deplete its energy. If either of these events occurs when a sensor in the WORKING status, some parts of its sensing area may no longer satisfy the required coverage. Therefore, we should provide a mechanism to detect the loss of area coverage and recover it as soon as possible. The heartbeat message monitoring approach [136] is not suitable in a wireless sensor architecture due to low power in sensors and the intrinsic distributed characteristic of sensor networks. The round-based sleeping configuration protocol will attempt to recover the area loss in the next round of sleeping scheduling [126]; however, timer T_{round} is a global parameter and not adaptive to recover a local area loss. Instead, we employ an adaptive sensor sleeping scheduling protocol by letting each sensor calculate its sleeping time locally and adaptively to build dependable sensor networks.

In the initial stage, all sensors are scheduled in the same way as the round-based protocol. After that, each sleeping sensor calculates its next wake-up time independently and sets a timer $T_{sleeping}$. When $T_{sleeping}$ times out, the sleeping sensor N_i wakes up and broadcasts a PROBE message to its neighbors. Each neighbor who is in the WORKING status and receives the PROBE message will return a STATUS message to the sender. The STATUS message contains not only a sensor's ID and status, but also its residual energy. Then N_i evaluates its sleeping eligibility according to the aforementioned sleeping candidate conditions. If N_i is not sleeping-eligible, it will set its status to WORKING and provides its sensibility. If it is sleeping-eligible, it will calculate its sleeping time. To do this, it generates a random sleeping time, compares this with the minimum remaining working time estimated from residual energy information collected from its neighbors, and sets the smaller of the two as its $T_{sleeping}$. With this adaptive sleeping protocol, each sensor will wake-up randomly and reevaluate its sleeping eligibility. If it is no longer sleeping-eligible due to its neighbors failing or running out of energy, it will switch to the WORKING status and recover the uncovered areas partially or completely. The recovery procedure is gradual, because sensors wake-up one by one. The PROBE message is constrained to reach one-hop neighbors, as broadcasting this message beyond one-hop during the sleeping period of neighbors may not exactly collects the required information while consuming too much energy. Therefore, the described adaptive sleeping configuration protocol is only appropriate to the sleeping candidate conditions for the BSM.

Note that, if we wish to detect an area coverage loss early, we must schedule sensors to scan their sensing disks more frequently, thus more energy will be consumed.

7.4 Simulations and Performance Evaluation

To evaluate and validate the capability of our proposed sleeping candidate conditions for two sensing models, the BSM and the CSM, and the SSCPs in coordinating sensors to sleep, we have implemented them in ns-2 [42] and conducted a simulation study. We implemented the sleeping candidate condition for the BSM with arc-coverage (*MpacB*) in the round-based SSCP, denoted as *SscpAc*, and in the adaptive SSCP, denoted as *SscpAcA*. We also denote the round-based SSCP with the sleeping candidate condition for the BSM with Voronoi diagram as *SscpVo* and the round-based SSCP with the sleeping candidate condition for the CSM as *SscpCo*.

7.4.1 Configuration Protocols for Comparison

According to the survey on coverage problems conducted in [19], we also evaluate as a baseline the performance of the sponsored sector (SS) eligibility rule proposed by Tian *et al.* [126] and Coverage Configuration Protocol (CCP) proposed by Wang *et al.* [133]. Both protocols are based on the BSM. The SS rule considers only the sensors inside the sensing radius of an evaluated sensor. The *CPP* determines a sensor's active eligibility by evaluating how the intersection points among sensing perimeters are covered inside the sensing disk of a considered sensor. The *CPP* states that when a sensor receives a message, it evaluates its working eligibility. It returns to the SLEEP state if not eligible. This is not correct as it will produce covered area reduction as a sensor sleeps prematurely. In our experiments, we make some corrections to ensure the *CPP* to preserve the covered area. A probabilistic distributed detection model which is similar as the CSM has been exploited by Xing *et al.* [134] with a Coordinating Grid (Co-Grid) protocol in the sleeping configuration. It divides the deployment region into grids and assumes there is a data fusion center in each grid. However, it is not fully localized, and fixing the location of the fusion center is not practical when sensors are randomly deployed. Therefore, it is not selected as a protocol for comparison. To evaluate the effectiveness of these distributed protocols in negotiating compensative sensors, we also construct a centralized algorithm with global coordination, denoted as *Central*, in which the same sleeping candidate condition as that of the *SscpCo* is implemented.

7.4.2 Parameters Setting

The deployed sensing area is 50m×50m [126, 139, 143]. Sensors are scattered in this area with a uniform distribution. $\alpha = 1$ and $\beta = 3$. The timer parameters are set as follows: $T_{round} = 100$s; $T_{hello-i}$ and $T_{hello-ii}$ are uniformly distributed between 0 and 0.3s; T_{win-i} and T_{win-ii} are set to 0.4s; T_{wait} = 2s. The power consumptions of Tx (transmit), Rx (receive), Idle, and Sleeping modes are 60mW, 12mW, 12mW, and 0.03mW, respectively [139]. The initial energy reserved for each sensor is uniformly distributed between 14J and 20J. As all simulated protocols need sensors' location information, the energy consumption for the GPS or other localization protocols is omitted. The default communication radius cr is 20m, the number of deployed sensors is 100, the coverage degree for the BSM is 1, the coverage requirement ϵ_s for the CSM is 0.001 (i.e., the ensured-sensibility radius is 10m), and the normalized deviation of location error is 0, unless specified. The granularity of the virtual square grid for the *SscpCo* is 2m [134, 136]. We also assume that there is no packet loss during simulation. All the results quoted were obtained from an average of 20 simulation runs.

7.4.3 Experimental Results and Discussions

In this subsection, we present the experimental results from different aspects: communication radius, number of deployed sensors, energy consumption, sensor redundancy, sensibility threshold, loss of area coverage, sensitivity to sensor failures, and network lifetime.

Sleeping Sensor vs. Communication Radius

Since the neighboring information is shared by broadcasting messages, the communication radius should affect the number of neighbors, and thus impact the percentage of sleeping sensors. Figure 7.14 shows the variation of the percentage of sleeping sensors with the communication radius without and with location error. As the *SS*, the *SscpAc*, and the *SscpVo* utilize one-hop neighbors, we also implemented the *CCP* with only one-hop neighbors for

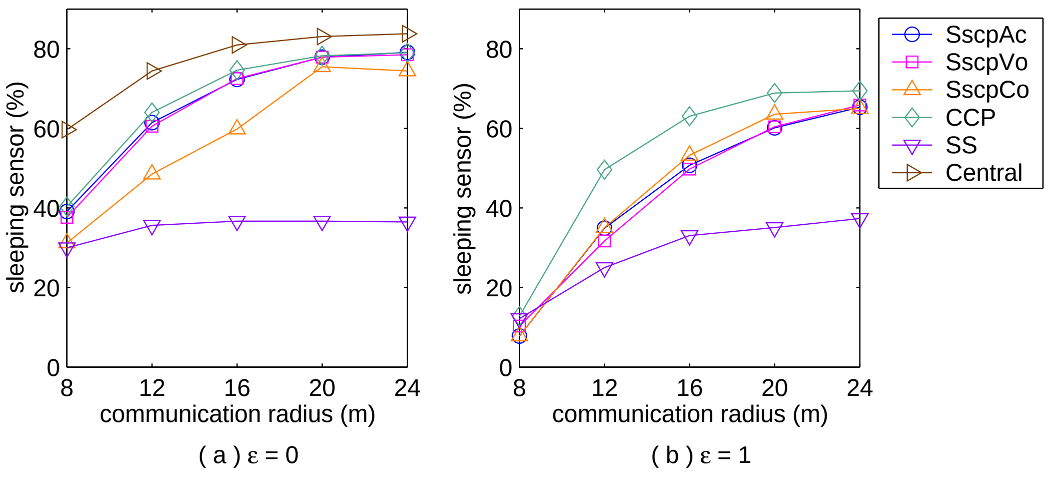


Figure 7.14: Percentage of sleeping sensors vs. communication radius cr.

comparison.

When we increase the communication radius, a sensor will identify more adjacent sensors. If a sensor has more neighbors, its responsible sensing area is more likely to be covered by its neighbors. As a result, more sensors will be sleeping-eligible. However, if we increase the communication radius further, the performance of all the protocols tends to be saturated. In addition, without location error (Figure 7.14(a)), the *SS* reaches saturation when $cr = sr$; for other protocols, the saturation condition is $cr = 2 \cdot sr$. This is because the *SS* only considers the neighbors which are in the sensing radius of a sensor as its sensing sponsors. Although there are some other sensors providing sensing sponsorship, it ignores them, leading to lower percentage of sleeping-eligible sensors. The performance of the *SscpAc*, the *SscpVo*, and the *CCP* are almost the same. However, the *SscpCo* achieves some lower performance, especially when cr is small. We know that a sensor with the CSM takes much more sensing responsibility than that with the BSM. Hence, in the *SscpCo* a sensor needs to scan its one-hop neighbors' Voronoi cells; while in the BSM, a sensor only needs to evaluate its own sensing disk which is much smaller. A larger responsible area implies that a sensor will get a smaller opportunity to be sleeping-eligible. This effect becomes more significant when the number of neighbors is small. With location error (Figure 7.14(b)), all distributed protocols identify fewer sleeping-eligible sensors. Although the performance of the *CCP* is higher than those of other protocols, the cost it must to pay is higher loss of area coverage, which is shown in Figure 7.21 later.

Compared with the result achieved by the *Central* algorithm, all the evaluated distributed protocols cannot identify sleeping candidates efficiently with a small cr. Even with a large cr, there is still a performance gap between centralized and distributed algorithms. The *SscpAc* performs the same as the *SscpVo*. This can be explained by that both are implemented in the round-based SSCP and they are equivalent in identify sleeping-eligible sensors.

Due to the sharing characteristic of the wireless transmission medium, the probability of packet collision is also increased when there is a long communication radius. Consequently, the number of neighbors discovered through the broadcasting approach may decrease. Linking this with the fact that a long communication range consumes more energy, we should choose an appropriate communication radius to enable acquisition of adequate neighboring information with as little energy consumption as possible while achieving the maximal percent of sleeping sensors. Therefore, for the following experiments we set $cr = 20\text{m}$.

Number of Working vs. Deployed Sensors

Figure 7.15 shows a plot of the number of working sensors as a function of the number of deployed sensors. A good sleeping configuration protocol should keep the number of working sensors irrespective of the number of deployed

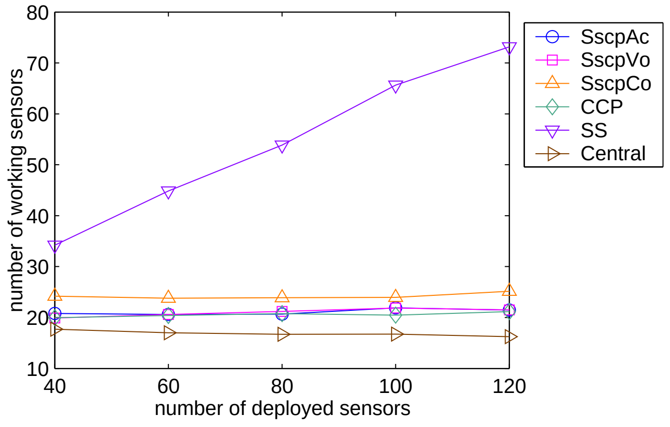


Figure 7.15: Number of working vs. deployed sensors.

sensors if the deployed sensors have covered the interested area. The simulation results show that all protocols engage this property except the *SS*. The distributed configuration protocol of the *SS* changes a sensor's status to WORKING as long as a sensor evaluates itself to be not sleeping-eligible, and does not send out a message to remind other sensors. However, other distributed configuration protocols allow sensors to change their status between READY-TO-WORKING and READY-TO-SLEEPING for catching each opportunity to go into sleeping. Another observation for a constrained deployment region is that the marginal sensors are almost always in working status with the *SS* as it does not deal with this case. The *CCP* additionally considers the intersection points between a sensing perimeter and the boundary of a deployment region. In our proposed sleeping candidate conditions, the *SscpAc* only employs the SSAs in a deployment region. In the *SscpVo* and the *SscpCo*, as the Voronoi cells are all finite, the boundary effect is automatically removed.

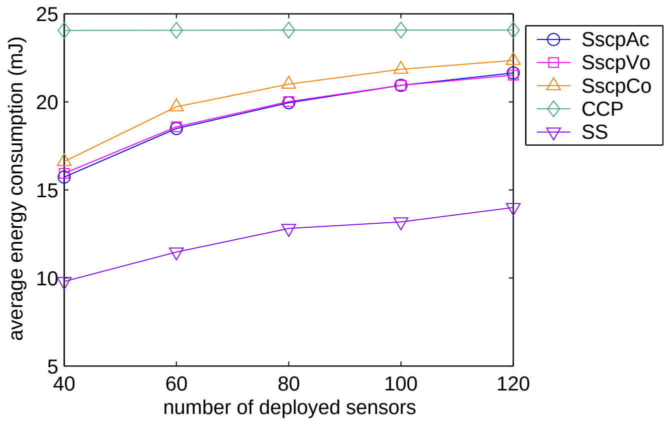


Figure 7.16: Energy consumption for configuration.

Energy Consumption for Configuration

Scheduling sensors is to conserve energy consumption; however, sleeping configuration itself consumes energy in transmission HELLO and other coordination messages. Therefore, we should also evaluate its energy cost. Figure 7.16 shows average energy consumption for each sensor in the configuration phase. The energy cost for the *SS* is the least, as it omits the fluctuation phase between READY-TO-WORKING and READY-TO-SLEEPING. The *SscpAc* and the *SscpVo* spends the same energy as they are implemented in the same configuration protocol. The *SscpCo* spends more energy than the *SscpAc* and the *SscpVo* do, which is obvious as the former needs to collect two-hop neighbors' information. In the *CPP*, a sensor engages two timers: JOIN and WITHDRAW, before sending out messages. As the energy consumptions for Tx, Rx, and Idle are comparable, the energy consumptions in these idle durations dominate that for configuration, thus it expends more energy than other protocols. However, one advantage is that its energy consumption does not change with the number of deployed sensors.

Sensor Redundancy

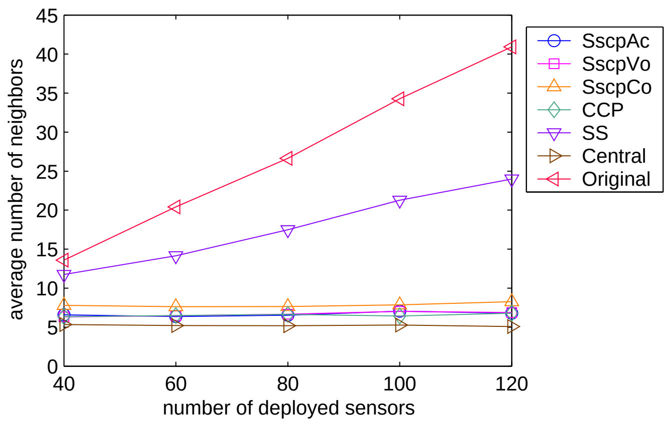


Figure 7.17: Average number of neighbors.

The average number of each sensor's communication neighbors and the distribution of field sensibility are two tractable measures for sensor redundancy, which are investigated in Figure 7.17 and Figure 7.18, respectively. Figure 7.17 shows how sleeping configuration reduces the average number of neighbors that each sensor perceives. Here the *Original* denotes the case that all deployed sensors are in working with no sleeping configuration. The effects of reducing the number of neighbors are two-fold. On the one hand, more sensors are scheduled to sleep thus prolonging system lifetime, and decreasing the opportunity of packet collision. On the other hand, the redundancy of sensors is also cut down, which may weaken the ability of FT.

The sensibility distributions for the original deployment and after configuration are shown in Figure 7.18. The sensibility calculations for the BSM and the CSM are different. In the CSM, the sensibility is additive, while in the BSM, it is not. Without configuration, the center of the sensibility distribution is far away from the desired sensibility $\epsilon_s = 0.001$, which results in depleting the energy of sensors too quickly. The *CCP*, the *SscpAc*, and the *SscpVo* makes most sensibility too close to the sensibility threshold, which positions them on

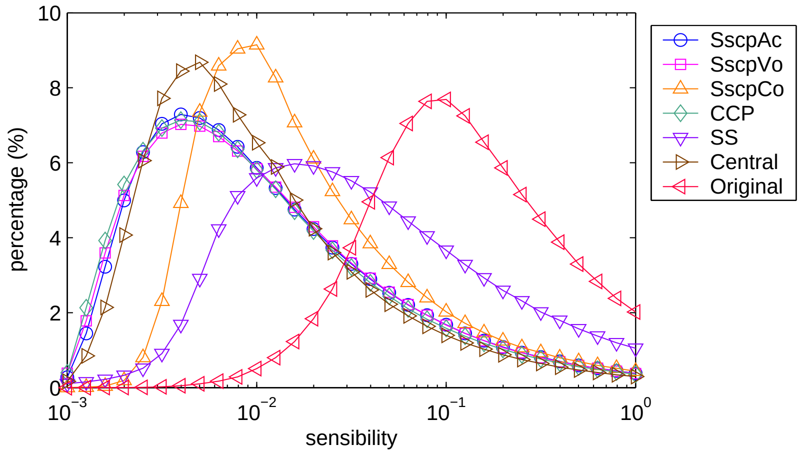


Figure 7.18: Field sensibility distribution.

a disadvantage situation for FT. Nevertheless, the *SscpCo* balances these two requirements: prolonging system lifetime and achieving FT. Therefore, the center of its sensibility distribution is in a moderate area.

Sleeping Sensors vs. Sensibility Threshold

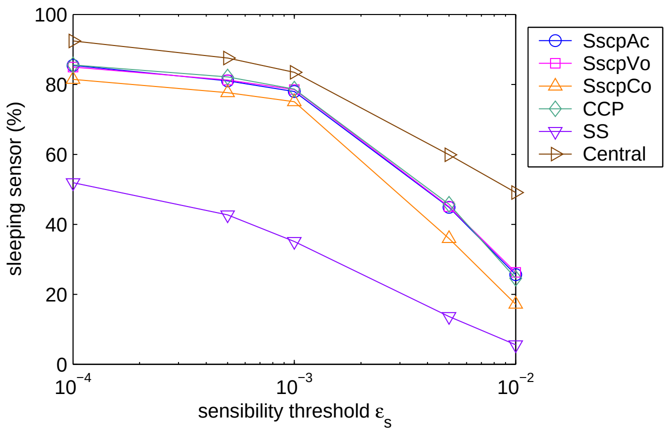


Figure 7.19: Percentage of sleeping sensors vs. sensibility threshold ϵ_s.

Figure 7.19 shows how the percentage of sleeping sensors changes with the sensibility threshold ϵ_s. If we enhance the coverage requirement in the

CSM (equivalent to decreasing the sensing radius in the BSM), the sleeping percentage is decreased. The result confirms that, when we demand higher event sensibility, more sensors must be working and fewer sensors are granted the opportunity to sleep.

Loss of Area Coverage

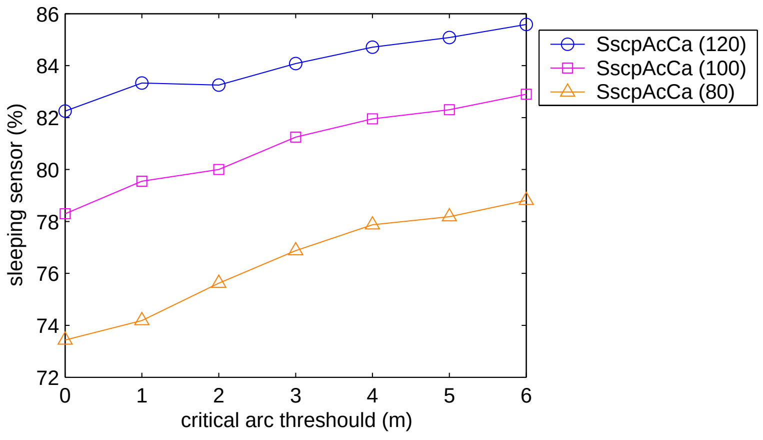


Figure 7.20: Percentage of sleeping sensor with *SscpAcCa*.

Table 7.1: Percentage of area coverage loss (%) with *SscpAcCa*.

	0	1	2	3	4	5	6
SscpAcCa (80)	0	0	0.1	0.2	0.6	0.8	1.1
SscpAcCa (100)	0	0	0.1	0.2	0.5	0.9	1.4
SscpAcCa (120)	0	0	0.1	0.2	0.5	0.8	1.5

The *SscpAcCa* is a round-based SSCP with the extension of critical arc to the sleeping candidate condition for the BSM with arc-coverage. It is a trade-off between loss of area coverage and an increase of sleeping sensors. Figure 7.20 shows how the percentage of sleeping sensors vary with the critical arc threshold when the number of deployed sensors are 80, 100, and 120, respectively; the corresponding coverage area losses are given in Table 7.1.

These results confirm that our simplified area loss estimation algorithm keeps the area loss at a low level while increasing the percentage of sleeping sensors.

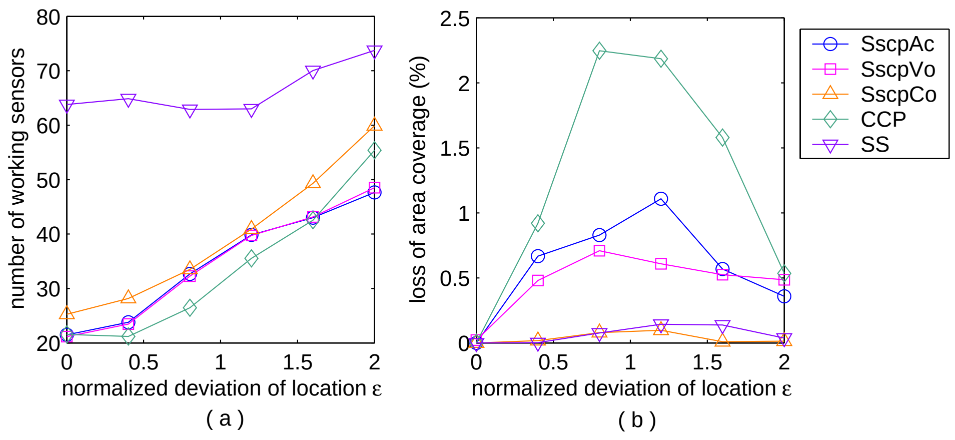


Figure 7.21: Deviation of location.

Figure 7.21 shows the effect of the location error on the loss of area coverage. With large deviations, fewer coverage sponsors are identified; therefore, the number of WORKING sensors of all sleeping scheduling protocols increase with the normalized deviation of location. As the *SS* only considers coverage sponsors in a sensor's sensing disk, it keeps enough redundant sensors to tolerate location error, and its loss of area coverage is almost 0, as shown in Figure 7.21(b). The *CCP* identifies sleeping-eligible sensors effectively; however, it does not take the location error into consideration. Therefore, although its number of WORKING sensors is the least when the normalized deviation of location is not very large, its loss of area coverage is the largest. The *SscpAc* and the *SscpVo*, on the other hand, reduce the loss of area coverage by allowing a few more sensors to work. The *SscpCo* also performs well in tolerating location error. One interesting observation is that the loss of area coverage will be decreased when the normalized deviation of location becomes larger. This reduction is a result of more working sensors. Even when the number of WORKING sensors in the *SscpAc* or the *SscpVo* is less than that in the *CCP*, the loss of area coverage in the *SscpAc* or the *SscpVo* is still less than that in the *CCP* with large deviation of location.

Sensitivity to Sensor Failures and Network Lifetime

To simulate failure due to causes other than energy depletion, such as destruction, malfunction, etc., we assume failures strike sensors according to an exponential distribution. The MTTF is set between 1000s and 5000s.

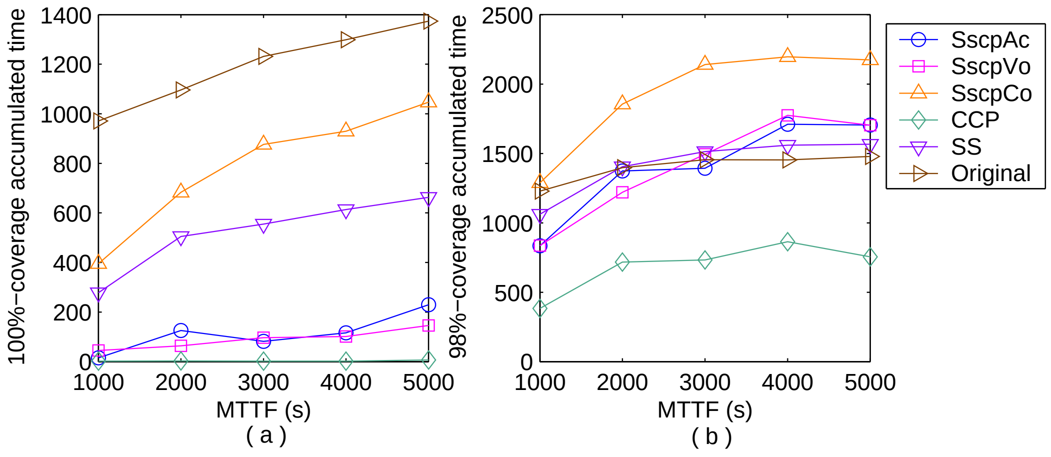


Figure 7.22: χ-coverage accumulated time vs. MTTF when $\epsilon = 1$.

The simulation results are shown in Figures 7.22 and 7.23 when the normalized deviation of location error $\epsilon = 1$. The data are given with the χ-*coverage accumulated time*, defined as the total time during which χ or more percentage of the original covered area still satisfies the coverage threshold. All χ-coverage accumulated times increase with the MTTF. If the original covered area should supply 100%-coverage as long as possible, the deployment of sensors should maintain as much redundancy as possible. Thus, the original deployment without sleeping configuration achieves the best performance, shown in Figure 7.22(a). The performance of the *SscpCo* is quite good as it exploits the collaboration between neighboring sensors and provides proper redundancy. The underestimation of sleeping-eligible sensors in the *SS* makes it in an advantage to provide 100%-coverage under location errors and sensor failures. Although the *SscpAc* and the *SscpVo* take the location error into consideration, they do not provide enough redundancy to tolerate sensor failures. Therefore, their performance in 100%-coverage is poor. The *CCP* almost cannot achieve 100%-coverage due to location error. This confirms the results of sensibility distribution in Figure 7.18. When we decrease χ a little to 98%, the advantage of our *SscpCo* is clearly revealed, shown in Figure 7.22(b). In this case, even the *SscpAc* and the *SscpVo* will perform comparably to the *Original* and the *SS*. This can be explained by the extended system lifetime by sensor configuration, which are shown in Figure 7.24(a) when the MTTF is 4000s. However, the *CCP* still does not provide acceptable performance.

To provide FT for coverage preservation, three approaches are investigated. The first one is the adaptive sleeping configuration protocol *SscpAcA*. The randomly generated sleeping time for the *SscpAcA* follows an exponential distribution with a mean equal to T_{round} for comparison with the round-based protocol. But all generated variables which are greater than $2T_{round}$ will be rounded down to $2T_{round}$. The second one is that we initially specify $(k+1)$-coverage, which provides one more coverage degree than the design requirement. The last approach is that we configure the sleeping sensors with short communication radius to keep sensor redundancy. In this specific simulation the latter two approaches are denoted as *SscpAc/2* and *SscpAc/R=12m*, respectively.

From Figure 7.23, we observe that all three FT approaches improves the 100%-coverage accumulated time. The *SscpAc/2* and the *SscpAc/R=12m* display similar performance on 100%-coverage, while the *SscpAcA* performs worse than those two. However, due to more energy consumption, their performance will be comparable to that of the *SscpAc* on 98%-coverage. Figure 7.24(b)

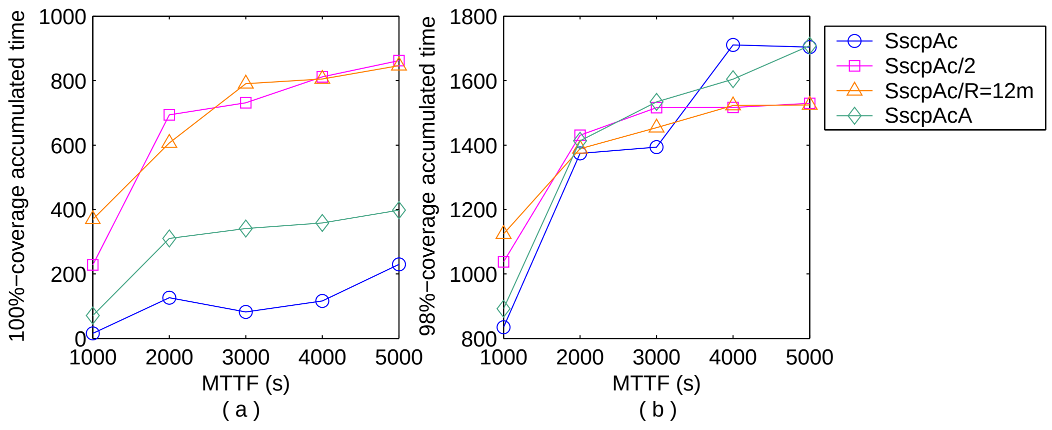


Figure 7.23: χ-coverage accumulated time vs. MTTF with FT approaches when $\epsilon = 1$.

shows their corresponding lifetime when the MTTF is 4000s.

Figure 7.24 shows the lifetime of sensors with different configuration protocols. The *CPP* extends the network lifetime most effectively because it does not take FT and location error into consideration and keeps the fewest sensors operating at any given time. The *Original* and the *SS* display a quicker decrease in the percentage of live sensors with time. The lifetime of the *SscpAc*, the *SscpVo*, and the *SscpCo* is somewhat less than that of the *CCP*; however, it is still much larger than that of the *SS*.

Taking the results in Figures 7.22, 7.23, and 7.24 together, we observe that there are three effective approaches to build a dependable sensor network. The *SscpAc/2* and the *SscpAc/R=12m* display similar performance; therefore, decreasing the communication radius or increasing the coverage degree is equivalent to providing FT, which is the first approach. Detecting sensor failures and recovering the area loss as quickly as possible is the second one, such as the *SscpAcA*. Exploiting the cooperation between neighboring sensors is the third one, such as the *SscpCo* with the CSM. We also observe that the cost we must pay for FT with the BSM is more energy consumption, thus

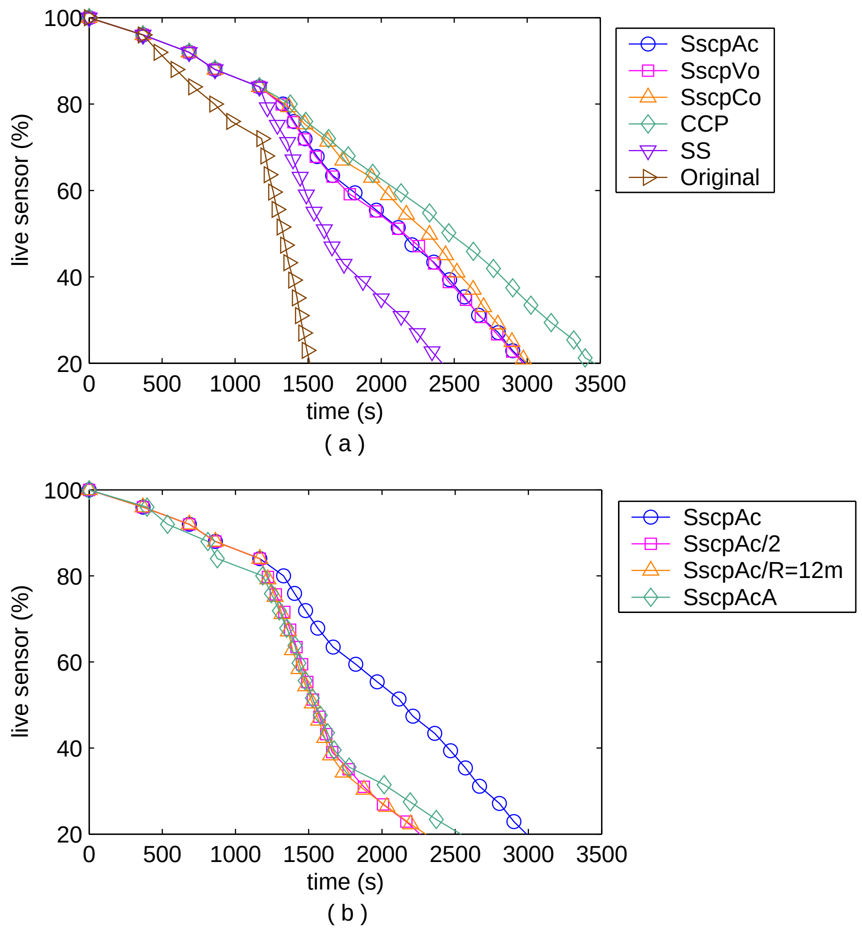


Figure 7.24: Percentage of live sensors vs. time when the MTTF is 4000s and $\epsilon = 1$.

shorter area monitoring time.

7.5 Summary

This chapter exploits problems of energy conservation and FT while maintaining desired coverage and network connectivity with location error in wireless sensor networks. Two sensing models are investigated: the BSM and the CSM. We develop three sleeping candidate conditions: one for the BSM with arc coverage, one for the BSM with Voronoi diagram, and one for the CSM. After that two distributed and localized sleeping configuration protocols (SSCPs) are developed, round-based and adaptive sleeping, which effectively identify redundant sensors and coordinates them to sleep for saving energy by exploiting the cooperation between adjacent sensors. Moreover, an adequate sensor redundancy is still kept to tolerate sensor failures and energy depletions. Finally, our sleeping candidate conditions integrate the sensing coverage requirement with the network connectivity, which results in the network still being connected after sleeping-eligible sensors turn off their communication devices. Our results show that there exists a trade-off among network lifetime, sensing coverage, and FT, which varies between different configuration protocols. Three effective approaches to build dependable wireless sensor networks are suggested: increasing the required degree of coverage or reducing the communication radius during sleeping configuration, configuring sensor sleeping adaptively, and utilizing the cooperation between neighboring sensors.

□ **End of chapter.**

Chapter 8

Conclusions and Future Directions

In this thesis, we investigate FT, performance, and reliability in wireless infrastructure networks (illustrated by wireless CORBA) and ad hoc sensor networks as their components are failure-prone. However, since the missions of these two types of wireless networks are different, we discuss their FT, performance, and reliability in different approaches.

First, on the basis of FT–CORBA and wireless CORBA specifications, we build a FT wireless CORBA with message logging and checkpointing. It employs both the quasi-sender-based and the receiver-based message logging methods. The protocol tolerates MH disconnection, MH crash, and AB crash. It chooses the storage available at AB as stable storage to log messages and save checkpoints. To tolerate AB crash, it replicates an AB's state on the previous AB for each MH. It also engages the handoff mechanism as a means to recover from AB crash. An opportunity for further research exists, such as optimistic message logging for wireless networks to reduce the block time introduced by pessimistic logging, hybrid checkpointing for wired and wireless networks to exploit different characteristics, FT broadcast/multicast protocols with MH handoff, mobile database recovery at HLA, fault detectors and consensus problems in malicious environments, etc.

After that, we analyze expected message sojourn time at AB in the presence of MH failures and handoffs with five message scheduling strategies: the basic queueing model, the static and the dynamic processor-sharing models, the cyclic polling model, and the feedback model. The expected message sojourn times are derived under steady state. Analytical and simulation results show that the basic model and the static processor-sharing dispatch model demonstrate the worst performance. The other three models may be suitable for applications as the dispatch strategy for AB; however, the runtime environment determines which one should be implemented. It would be interesting to derive analytical results for the left three models: the basic model, the cyclic polling model, and the feedback model, to generalize the exponentially distributed message arrival interval and service time, to examine the performance in the presence of additional AB failure, to assess the load of AB imposed by FT protocols, and to evaluate the average recovery time due to scattered message logs.

Third, we study the program execution time with three checkpointing strategies in wireless networks: deterministic checkpointing, random checkpointing, and time-based checkpointing. The termination requirement for a program at MH is now changed from time used in the literature for single host to the number of computational messages that it should receive. We assume that MH and wireless link failure intervals, message arrival interval, and handoff interval are r.v.s with exponential distributions. We derive the LST of the c.d.f. of the total program execution time and its expectation. We show that the performance of random checkpointing approach is more stable against varying parameter conditions. Based on our study, different checkpointing strategies, including the absence of checkpointing, can be engaged to achieve optimal performance under different mobile wireless network conditions. In our performance analysis model, we assume that after a failure the messages still arrive at MH as that in the normal situation with exponential distribution. But during the recovery interval (repair and rollback), messages are queued at AB. So after a failure, there should be a time interval in which the time between two successive message arrivals is not distributed exponentially. We should take this into consideration to construct a more realistic analysis model. Another potential area of research is to study the effect of wireless bandwidth and MH disconnection on program execution time.

Furthermore, we extend traditional reliability analysis of wired networks to wireless networks with imperfect components. The unique characteristic of wireless networks is handoff, which leads to different communication structures in which various types and numbers of components are engaged. Traditional reliability definition is not suitable to be employed in this situation, so we propose a new term, end-to-end mobile reliability, to describe reliability scenarios in wireless networks. We also identify the RI of each component with respect to the mobile reliability. Under different conditions, different components in wireless networks should be focused on to receive the highest reliability gain. Future work should involve development of end-to-end reliability evaluation for sensor networks, in which one end is composed of a cluster of sensors, thus the failure of one or more sensors may not cause the failure of event detection. Along with the reliability issue, message delay due to sensor failures should also be evaluated.

Finally, this thesis exploits energy conservation and FT while maintaining area coverage and network connectivity with location error in wireless ad hoc sensor networks. We investigate two sensing models, the BSM and the CSM, and develop three sleeping candidate conditions based on arc-coverage and

Voronoi diagram. The sleeping candidate conditions can be applied with heterogeneous sensors and with various coverage requirements. Then a distributed and localized sleeping configuration protocol (SSCP) is presented, which effectively identifies redundant sensors and coordinates them to sleep for saving energy by collecting one- or two-hop neighbors information. Three effective approaches to build dependable wireless sensor networks are suggested: increasing the required degree of coverage or reducing the communication radius during sleeping configuration, coordinating sensor sleeping adaptively, and utilizing the cooperation between neighboring sensors. A number of issues remain to be studied. We are investigating on relaxing the assumption of known location information and no packet loss in transmission. Finding a reliable path to report event detection to end-user and integrating sleeping configuration protocol with routing protocols and data delivery protocols are also very important to build practical wireless sensor networks.

Appendix A

List of Acronyms

AB Access Bridge

AP Access Point

CORBA Common Object Request Broker Architecture

ba basic dispatch

BSM Boolean Sensing Model

CCS Complete-Coverage Sponsor

c.d.f. cumulative distribution function

cp cyclic polling dispatch

CSM Collaborative Sensing Model

CSFS Collective-Sensor Field Sensibility

CSG Covered Sensing Angle

dc deterministic checkpointing

DOC Degree of Complete Coverage

dps	dynamic processor-sharing dispatch
edc	extended deterministic checkpointing
fb	feedback dispatch
FCFS	First-Come-First-Served
FT	Fault Tolerance or Fault-Tolerant
GIOP	General Inter-ORB Protocol
GTP	GIOP Tunneling Protocol
HLA	Home Location Agent
IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
i.i.d.	independent and identically distributed
LST	Laplace-Stieltjes Transform
MH	Mobile Host
MIOR	Mobile IOR
MPAC	Minimum Partial Arc-Coverage
MLE	Message Log Entry
MR	Mobile Reliability
MTTF	Mean Time To Failure
NAB	New AB

NSFS Neighboring-Sensor Filed Sensibility

OAB Old AB

OMG Object Management Group

ORB Object Request Broker

p.m.f probability mass function

rc random checkpointing

RI Reliability Importance

r.v. random variable

SH Static Host

SN Sequence Number

sps static processor-sharing dispatch

SSCP Sensibility-based Sleeping Configuration Protocol

SSA Sponsored Sensing Arc

SSG Sponsored Sensing Angle

SSR Sponsored Sensing Region

tc time-based checkpointing

Appendix B

List of Notations

α energy emitted by an event occurring at point y

β decaying factor of a sensing signal

γ_a AB's failure rate

γ_h HLA's failure rate

γ_l wireless link's failure rate

γ_m MH's failure rate

γ_s SH's failure rate

ϵ normalized deviation of location

ϵ_d maximum deviation of location

ϵ_s sensibility threshold

ϵ_n signal threshold

ζ_i degree of complete coverage of sensor N_i

η handoff completion rate

θ_{ij} sponsored sensing angle for sensor N_j to sensor N_i

ι checkpointing rate

κ MH's recovery rate

$\Lambda(t)$ MH's state at time t

λ message arrival rate

λ^* message arrival rate in the presence of wireless link failures

μ AB's message service rate

ν location-forwarding rate

ξ_{ij} minimum partial arc-coverage sponsored by sensor N_j to sensor N_i

$\pi_x(t)$ $\Pr\{\text{state } x\}$, $x \in \{a, b, \ldots, r\}$

ρ MH's handoff rate

ρ_a AB's traffic intensity

ρ_d expected traffic intensity of a message queue with the dynamic processor-sharing dispatch model

σ_{ij} direction of sensor N_i referred to sensor N_j

τ_{ij} sponsored sensing arc for sensor N_j to sensor N_i

υ switchover rate in the cyclic polling and feedback dispatch models

Φ sensor deployment region

$\phi_Z(s)$ LST of the c.d.f. of a r.v. Z

Ψ_i sensing disk of sensor N_i

Ω deployed sensor set in a sensor deployment region Φ

ω_{ij} covered sensing angle of sensor N_i by sensor N_j

A average effectiveness which is the ratio between the expected program execution time without and with failures, handoffs, and checkpoints

C total checkpointing time, i.e., $T_1^{(h,l)} + T_2^{(l)}$

c event of checkpointing

$CCS(i)$ complete-coverage sponsors of sensor N_i

cr communication radius of a sensor

D time requirement to dispatch a message at AB

d normalized distance

$d(N_i, N_j)$ Euclidean distance between sensors N_i and N_j

$E(Z)$ expectation of a r.v. Z

f event of MH failure

$G(s,n,v)$ $Q_1^n(s)\left[1 - \phi_v(Q_3(s))\sum_{i=0}^{n-1}\frac{(vQ_3(s))^i}{i!}\right]$

$G_Z(t)$ general c.d.f. of a r.v. Z

H MH's handoff time

h event of handoff

I number of received messages before taking a checkpoint with random checkpointing

$I_{R_i}(t)$ RI of component $i \in \{mh, ab, sh, hla\}$

L number of deliverable messages in the dispatch facility with the dynamic processor-sharing dispatch model

l event of wireless link failure

$MR_r(t)$ end-to-end MR for scheme $r \in \{ss, ms, sm, mm\}$

m number of MHs covered by an AB

$N_{dc}(n, u)$ number of checkpoints with deterministic checkpointing

N_i sensor i

$N(i)$ one-hop working neighbors of sensor N_i

$N_{rc}(n, p)$ number of checkpoints with random checkpointing

n number of computational messages that an MH should receive to complete its program

n_c number of component $c \in \{mh, sh\}$

p parameter of a geometric distribution

p_x stationary probability in state $x \in \{0, 1, 2\}$

$p(N_i, y)$ probability of point y covered by sensor N_i

Q rate matrix of a Markov process

$Q_1(s)$ $\frac{\lambda}{s+\gamma_1+\lambda+\rho-\rho\phi_{H(l)}(s+\gamma_1)}$

q_1 $Q_1(s)|_{s=0}$

$Q_2(s)$ $\frac{p\phi_C(s+\gamma_1)}{1-(1-p)Q_1(s)}$

q_2 $Q_2(s)|_{s=0}$

$Q_3(s)$ $s + \gamma_1 + \lambda + \rho - \rho\phi_{H^{(l)}}(s + \gamma_1)$

Q_3 $Q_3(s)|_{s=0}$

R repair time

R' total time between a failure and the instant that the program is ready to receive computational messages, i.e., $[R + T_3^{(l)} + T_4^{(h,l)}]^{(f)}$

$R_i(t)$ reliability of component i at time t, $i \in \{mh, ab, sh, hla\}$

S switchover time

$S_c(y)$ collective-sensor field sensibility for an event occurring at point y

$S_n^i(y)$ neighboring-sensor field sensibility perceived by sensor N_i for an event occurring at point p

s s-domain in LST

$s(N_i, y)$ sensibility of sensor N_i for an event occurring at a measuring point y

sr_i sensing radius of sensor N_i

sr_i^c collaborative-sensing radius of sensor N_i

sr_i^e ensured-sensing radius of sensor N_i

T_1 time to take a checkpoint on an MH

T_2 time to save a checkpoint on stable storage

T_3 time to retrieve a checkpoint to an MH

T_4 time to reload a checkpoint

T_z message sojourn time with the dispatch strategy $z \in \{ba, sps, dps, cp, fb\}$ at AB

U MH's recovery time

u number of received messages before taking a checkpoint with deterministic checkpointing

u' number of left messages after taking the last checkpoint interval with deterministic checkpointing

v time between two sequential checkpoints with time-based checkpointing

w number of program execution intervals with deterministic checkpointing

$X(n)$ total program execution time with n messages in the absence of MH and wireless link failures, handoffs, and checkpointings

x system communication state

Y time to the first MH failure

$Z^{(e)}$ total program execution time in the presence of events $e \subseteq \{f, l, h, c\}$ with the time requirement Z

z system failure state

Bibliography

[1] A. Acharya, B. Badrinath, and T. Imielinski. Checkpointing distributed applications on mobile computers. In *Proc. of the 3rd International Conference on Parallel and Distributed Information Systems*, pages 73–80, Austin, Texas, Sept. 1994.

[2] K. K. Aggarwal, J. S. Gupta, and K. B. Misra. A simple method for reliabiilty evaluation of a communication system. *IEEE Transactions on Communications*, 23(5):563–566, May 1975.

[3] J. H. Ahn and C. S. Hwang. Low-cost fault-tolerance for mobile nodes in mobile ip based systems. In *Proc. of the 15th International Parallel and Distributed Processing Symposium*, pages 508–513, San Francisco, California, Apr. 2001.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 102–114, Aug. 2002.

[5] S. Alagra, R. Rajagopalan, and S. Venkatesan. Tolerating mobile support station failures. In *Proc. of the 1st Conference on Fault Tolerant Systems*, pages 225–231, Madras, India, Dec. 1995.

[6] R. Alena, D. Evenson, and V. Rundquist. Analysis and testing of mobile wireless networks. In *Proc. of 2002 IEEE Aerospace Conference*, volume 3, pages 1131–1144, Big Sky, Montana, Mar. 2002.

[7] R. Alena, E. Yaprak, and S. Lamouri. Modeling a wireless network for international space station. In *Proc. of 2000 IEEE Aerospace Conference*, volume 1, pages 223–228, Big Sky, Montana, Mar. 2000.

[8] T. Altiok. Queueing modeling of a single processor with failures. *Performance Evaluation*, 9(2):93–102, 1988/89.

[9] L. Alvisi and K. Marzullo. Message logging: Pessimistic, optimistic, causal, and optimal. *IEEE Transactions on Software Engineering*, 24(2):149–159, Feb. 1998.

[10] ANSI/IEEE. ANSI/IEEE std. 802.11, 1999 edition. 1999.

[11] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, Sept. 1991.

[12] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan.-Mar. 2004.

[13] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. of IEEE Infocom 2003*, volume 3, pages 1713–1723, San Franciso, California, Mar. 2003.

[14] M. Barborak, M. Malek, and A. Dahbura. The consensus problem in fault-tolerant computing. *ACM Computing Surveys*, 25(2):171–220, June 1993.

[15] G. Bolch, S. Greiner, H. d. Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* John Wiely & Sons, New York, 1998.

[16] G. Cao and M. Singhal. Mutable checkpoints: A new checkpointing approach for mobile computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(2):157–172, Feb. 2001.

[17] B. Carbunar, A. Grama, and J. Vitek. Distributed and dynamic voronoi overlays for coverage detection and distributed hash tables in ad-hoc networks. In *Proc. of the 10th International Conference on Parallel and Distributed Systems*, pages 549–555, Newport Beach, California, July 2004.

[18] B. Carbunar, A. Grama, J. Vitek, and O. Carbunar. Coverage preserving redundancy elimination in sensor networks. In *Proc. of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 377–386, Santa Clara, California, Oct. 2004.

[19] M. Cardei and J. Wu. Energy-efficient coverage problems in wireless ad hoc sensor networks. *Journal of Computer Communications on Sensor Networks*, 2005. to be published.

[20] A. Cerpa and D. Estrin. ASCENT: Adaptive self-configuring sensor networks topologies. *IEEE Transactions on Mobile Computing*, 3(3):1–14, July 2004.

[21] K. M. Chandy, J. C. Browne, C. W. Dissly, and W. R. Uhrig. Analytic models for rollback and recovery strategies in data base systems. *IEEE Transactions on Software Engineering*, SE-1(1):100–110, Mar. 1975.

[22] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8:481–494, 2002.

[23] I.-R. Chen, B. Gu, S. George, and S.-T. Cheng. On failure recoverability of client-server applications in mobile wireless environments. *IEEE Transactions on Reliability*, 54(1):115–122, Mar. 2005.

[24] X. Chen and M. R. Lyu. Message logging and recovery in wireless CORBA using access bridge. In *Proc. of the 6th International Symposium on Autonomous Decentralized Systems*, pages 107–114, Pisa, Italy, Apr. 2003.

[25] X. Chen and M. R. Lyu. Performance and effectiveness analysis of checkpointing in mobile environments. In *Proc. of the 22nd Symposium on Reliable Distributed Systems*, pages 131–140, Florence, Italy, Oct. 2003.

[26] X. Chen and M. R. Lyu. Expected-reliability analysis for wireless CORBA with imperfect components. In *Proc. of the 10th International Symposium Pacific Rim Dependable Computing*, pages 207–215, Tahiti, French Polynesia, Mar. 2004.

[27] X. Chen and M. R. Lyu. Queueing analysis for access points with failures and handoffs of mobile stations in wireless networks. In *Proc. of 2004 IEEE Aerospace Conference*, volume 1, pages 1296–1304, Big Sky, Montana, Mar. 2004.

[28] X. Chen and M. R. Lyu. Analysis of program execution time based on various checkpointing strategies in mobile wireless environments. *IEEE Transactions on Mobile Computing*, 2005. submitted.

[29] X. Chen and M. R. Lyu. Ensuring area coverage with node scheduling in wireless sensor networks. 2005. submitted to the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems.

[30] X. Chen and M. R. Lyu. Reliability analysis for various communication schemes in wireless CORBA. *IEEE Transactions on Reliability*, 54(2):232–242, June 2005.

[31] X. Chen and M. R. Lyu. Sensibility-based sleeping configuration for wireless sensor networks. 2005. submitted to the 3rd ACM Conference on Embedded Networked Sensor Systems.

[32] E. G. Coffman. Waiting time distributions for processor-sharing systems. *Journal of the ACM*, 17(1):123–130, Jan. 1970.

[33] E. G. Coffman and L. Kleinrock. Feedback queueing models for time-shared systems. *Journal of the ACM*, 15(4):549–576, Oct. 1968.

[34] E. G. Coffman Jr and E. N. Gilbert. Optimal strategies for scheduling checkpoints and preventive maintenance. *IEEE Transactions on Reliability*, 39(1):9–18, Apr. 1990.

[35] F. Cristian and F. Jahanian. A timestamp-based checkpointing protocol for long-lived distributed computations. In *Proc. of the 10th Symposium on Reliable Distributed Systems*, pages 12–20, Pisa, Italy, Sept. 1991.

[36] H. Dai and R. Han. TSync: A lightweight bidirectional time synchronization service for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(1):125–139, Jan. 2004.

[37] B. Dimitrov, Z. Khalil, N. Kolev, and P. Petrov. On the optimal total processing time using checkpoints. *IEEE Transactions on Software Engineering*, 17(5):436–442, May 1991.

[38] W. Dotson and J. Gobien. A new analysis technique for probabilistic graphs. *IEEE Transactions on Circuits and Systems*, 26(10):855–865, Oct. 1979.

[39] A. Duda. The effects of checkpointing on program execution time. *Information Processing Letters*, 16:221–229, June 1983.

[40] M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Computing Surveys*, 34(3):375–408, Sept. 2002.

[41] W. Emmerich. *Engineering Distributed Objects.* John Wiley & Sons, Ltd, Chichester, 2000.

[42] K. Fall and K. Varadhan. *The ns manual*, Mar. 2005. http://www.isi.edu/nsnam/ns.

[43] W. K. Fuchs, N. Neves, and K.-F. Ssu. Dependable distributed and mobile computing - utilizing time to enhance recovery from failures. In D. R. Avresky, editor, *Dependable Network Computing*, chapter 14, pages 315–339. Kluwer Academic, 2000.

[44] S. Gadiraju and V. Kumar. Recovery in the mobile wireless environment using mobile agents. *IEEE Transactions on Mobile Computing*, 3(2):180–191, Apr.–June 2004.

[45] Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proc. of the 2nd ACM International Workshop on Wireless*

Sensor Networks and Applications, pages 108–114, San Diego, California, Sept. 2003.

[46] S. Garg, Y. Huang, C. Kintala, and K. S. Trivedi. Minimizing completion time of a program by checkpointing and rejuvenation. In *Proc. of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 252–261, Philadelphia, USA, May 1996.

[47] D. P. Gaver Jr. A waiting line with interrupted service, including priorities. *Journal of the Royal Statistical Society, Series B*, 24:73–90, 1962.

[48] J.-C. Geffroy and G. Motet. *Design of Dependable Computing Systems.* Kluwer Academic Publisers, Dordrecht, 2002.

[49] E. Gelenbe. On the optimum checkpoint interval. *Journal of the ACM*, 26(2):259–270, Apr. 1979.

[50] E. Gelenbe and D. Derochette. Performance of rollback recovery systems under intermittent failures. *Communications of the ACM*, 21(6):493–499, June 1978.

[51] V. Grassi, L. Donatiello, and S. Tucci. On the optimal checkpointing of critical tasks and transaction-oriented systems. *IEEE Transactions on Software Engineering*, 18(1):72–77, Jan. 1992.

[52] P. Hall. *Introduction to the Theory of Coverage Processes.* John Wiley & Sons, New York, 1988.

[53] B. R. Haverkort. *Performance of Computer Communication Systems: A Model-Based Approach.* John Wiely & Sons, Chichester, 1998.

[54] B. R. Haverkort, R. Marie, G. Rubino, and K. Trivedi, editors. *Performability Modelling: Techniques and Tools.* John Wiely & Sonsy, Chichester, 2001.

[55] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of the 33rd Hawaii International Conference on System Sciences*, pages 1–10, Maui, Hawaii, Jan. 2000.

[56] H. Higaki and M. Takizawa. Checkpoint-recovery protocol for reliable mobile systems. In *Proc. of the 17th Symposium on Reliable Distributed Systems*, pages 93–99, West Lafayette, Indiana, Oct. 1998.

[57] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Transactions on Computers*, 3(8):57–66, Aug. 2001.

[58] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *Proc. of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications*, pages 115–121, San Diego, California, Sept. 2003.

[59] J. Jiang and W. Dou. A coverage-preserving density control algorithm for wireless sensor networks. In *Proc. of the 3rd International Conference on AD-HOC Networks and Wireless*, pages 42–55, Vancouver, British Columbia, July 2004.

[60] J. Jing, A. Helal, and A. Elmagarmid. Client-server computing in mobile environments. *ACM Computing Surveys*, 31(2):117–156, June 1999.

[61] D. B. Johoson. *Distributed System Fault Tolerance Using Message Logging and Checkpointing.* PhD thesis, Rice University, Dec. 1989.

[62] R. Jokl and S. Racek. C-Sim version 5.1. Technical Report DCSE/TR-2003-17, Univ. of Wet Bohemia in Pilsen, May 2003.

[63] T.-Y. Juang. Crash recovery for distributed mobile computing systems. *IEICE Transactions on Fundamentals of Electronic, Communications, and Computer Science*, E84-A(2):668–674, Feb. 2001.

[64] W.-J. Ke and S.-D. Wang. Reliability evaluation for distributed computing networks with imperfect nodes. *IEEE Transactions on Reliability*, 46(3):342–349, Sept. 1997.

[65] S. A. Khan and M. I. Abd-El-Barr. On the use of fuzzy logic in a hybrid scheme for tolerating mobile support station failure. In *Proc. of the IEEE International Conference on Fuzzy Systems*, pages 717–722, Honolulu, Hawaii, May 2002.

[66] G. Khanna, S. Bagchi, and Y.-S. Wu. Fault tolerant energy aware data dissemination protocol in sensor networks. In *Proc. of the 2004 International Conference on Dependable Systems and Networks*, pages 739–748, Florence, Italy, June 2004.

[67] L. Kleinrock. *Queueing Systems, Volumn I: Theory.* John Wiely & Sons, New York, 1975.

[68] L. Kleinrock. *Queueing Systems, Volumn I: Computer Applications.* John Wiely & Sons, New York, 1976.

[69] L. Kleinrock and R. R. Muntz. Processor sharing queueing models of mixed scheduling disciplines for time shared systems. *Journal of the ACM*, 19:464–482, July 1972.

[70] C. M. Krishna, K. G. Shin, and Y.-H. Lee. Optimization criteria for checkpoint placement. *Communications of the ACM*, 27(10):1008–1012, Oct. 1984.

[71] P. Krishna. *Performance Issues in Mobile Wireless Networks.* PhD thesis, Texas A&M University, Aug. 1996.

[72] P. Krishna, N. H. Vaidya, and D. K. Pradhan. Recovery in distributed mobile environments. In *Proc. of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 83–88, Princeton, New Jersey, Oct. 1993.

[73] P. Kubat. Estimation of reliability for communication/computer networks – simulation/analytic approach. *IEEE Transactions on Communications*, 37(9):927–933, Sept. 1989.

[74] V. G. Kulkarni, V. F. Nicola, and K. S. Trivedi. Effects of checkpointing and queueing on program performance. *Communications in Statistics - Stochastic Models*, 6(4):615–648, 1990.

[75] W. Kuo and M. J. Zuo. *Optimal Reliability Modeling: Principles and Applications.* John Wiley & Sons Inc., New Jersey, 2003.

[76] P. L'Ecuyer and J. Malenfant. Computing optimal checkpointing strategies for rollback and recovery systems. *IEEE Transactions on Computers*, 37(4):491–496, Apr. 1988.

[77] P. A. Lee and T. Anderson. *Fault Tolerance – Principles and Practice*, volume 3 of *Dependable Computing and Fault-Tolerant Systems.* Springer-Verlag/Wien, New York, second edition, 1990.

[78] S. M. Lee and D. H. Park. An efficient method for evaluating network-reliability with variable link-capacities. *IEEE Transactions on Reliability*, 50(4):374–451, Dec. 2001.

[79] X.-Y. Li, P.-J. Wan, and O. Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52(6):753–762, June 2003.

[80] C.-M. Lin and C.-R. Dow. Efficient checkpoit-based failure recovery techniques in mobile computing systems. *Journal of Information Science and Engineering*, 17(4):549–573, July 2001.

[81] Y. Ling, J. Mi, and X. Lin. A variational calculus approach to optimal checkpoint placement. *IEEE Transactions on Computers*, 50(7):699–708, July 2001.

[82] B. Liu and D. Towsley. A study of the coverage of large-scale sensor networks. In *Proc. of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 475–483, Fort Lauderdale, Florida, Oct. 2004.

[83] J. Liu, X. Koutsoukos, J. Reich, and F. Zhao. Sensing field: Coverage characterization in distributed sensor networks. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages V–173–176, Hong Kong, Apr. 2003.

[84] M. R. Lyu, editor. *Software Fault Tolerance.* John Wiley & Sons Ltd., 1995.

[85] M. R. Lyu, editor. *Handbook of Software Reliability Engineering.* IEEE Computer Society Press and McGraw-Hill Book Company, 1996.

[86] M. R. Lyu, X. Chen, and T.-Y. Wong. Design and evaluation of a fault-tolerant mobile-agent system. *IEEE Intelligent Systems*, 19(5):32–38, Sept./Oct. 2004.

[87] D. Manivannan and M. Singhal. Quasi-synchronous checkpointing: Models, characterization, and classification. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):703–713, July 1999.

[88] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. of the 2nd ACM International Conference on Embedded Networked Sensor Systems*, pages 39–49, Baltimore, Maryland, Nov. 2004.

[89] S. Megerian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Worst and best-case coverage in sensor networks. *IEEE Transactions on Mobile Computing*, 4(1):84–92, Jan./Feb. 2005.

[90] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak. Exposure in wireless sensor networks: Theory and practical solutions. *Wireless Networks*, 8:443–454, 2002.

[91] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of IEEE Infocom 2001*, pages 1380–1387, Anchorage, Alaska, Apr. 2001.

[92] L. E. Moser, P. M. Melliar-Smith, and P. Narasimhan. A fault tolearnce framework for CORBA. In *Proc. of the 29th International Symposium on Fault-Tolerant Computing*, pages 150–157, Madison, Wisconsin, June 1999.

[93] V. A. Netes and B. P. Filin. Consideration of node failures in network-reliability calculation. *IEEE Transactions on Reliability*, 45(1):127–128, Mar. 1996.

[94] N. Neves and W. K. Fuchs. Adaptive recovery for mobile environments. *Communications of the ACM*, 40(1):68–74, Jan. 1997.

[95] V. F. Nicola. A single server queue with mixed types of interruptions. *Acta Informatica*, 23:465–486, 1986.

[96] V. F. Nicola. Checkpointing and the modeling of program execution time. In M. R. Lyu, editor, *Software Fault Tolerance*, chapter 7, pages 167–188. John Wiley & Sons Ltd., 1995.

[97] V. F. Nicola, V. G. Kulkarni, and K. S. Trivedi. Queueing analysis of fault-tolerant computer systems. *IEEE Transactions on Software Engineering*, 13:363–375, Mar. 1987.

[98] R. Nunez-Queija. Sojourn times in a processor sharing queue with service interruptions. *Queueing Systems*, 34:351–386, 2000.

[99] Object Management Group. The Common Object Request Broker: Architecture and specification, 3.0.3 edition. *OMG Document formal/04-03-01*, Mar. 2004.

[100] Object Management Group. Wireless access and terminal mobility in CORBA, 1.1 edition. *OMG Document formal/04-04-02*, Apr. 2004.

[101] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Ltd, Chichester, 2000.

[102] T. Park, N. Woo, and H. Y. Yeom. An efficient recovery scheme for mobile computing environments. In *Proc. of the 8th International Conference on Parallel and Distributed Systems*, pages 53–60, KyongJu City, Korea, June 2001.

[103] T. Park and H. Y. Yeom. An asynchronous recovery scheme based on optimistic message logging for the mobile computing systems. In *Proc. of the 20th International Conference on Distributed Computing Systems*, pages 436–443, Taipei, Taiwan, Apr. 2000.

[104] J. S. Plank and M. G. Thomason. The average availability of uniprocessor checkpointing systems, revisited. Technical Report UT-CS-98-400, Univ. of Tennessee, Aug. 1998.

[105] D. K. Pradhan, P. Krishna, and N. H. Vaidya. Recoverable mobile environment: Design and trade-off analysis. In *Proc. of the 26th International Symposium on Fault-Tolerant Computing*, pages 16–25, Sendai, Japan, June 1996.

[106] R. Prakash and M. Singhal. Low-cost checkpointing and failure recovery in mobile computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):1035–1048, Oct. 1996.

[107] P. J. Rasch. A queueing theory study of round-robin scheduling of time-shared computer systems. *Journal of the ACM*, 17(1):131–145, Jan. 1970.

[108] R. Ruggaber and J. Seitz. Using CORBA applications in nomadic environments. In *Proc. of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, pages 161–170, Monterey, California, Dec. 2000.

[109] R. Ruggaber and J. Seitz. A transparent network handover for nomadic CORBA users. In *Proc. of the 21st International Conference on Distributed Computing Systems*, pages 499–506, Phoenix, Arizona, Apr. 2001.

[110] H. Rutagemwa and X. Shen. Modeling and analysis of WAP performance over wireless links. *IEEE Transactions on Mobile Computing*, 2(3):221–232, July–Sept. 2003.

[111] R. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package.* Kluwer Acadmedic Publishers, Boston, 1996.

[112] J. L. Schiff. *The Laplace Transform: Theory and Applications.* Springer, New York, 1999.

[113] J. Shaio. A family of algorithms for network reliability problems. In *Proc. of 2002 IEEE International Conference on Communications*, volume 4, pages 2167–2173, New York, Apr. 2002.

[114] Y. Shang, H. Shi, and A. A. Ahmed. Performance study of localization methods for ad-hoc sensor networks. In *Proc. of the 1st International Conference on Mobile Ad-hoc and Sensor Systems*, pages 184–193, Fort Lauderdale, Florida, Oct. 2004.

[115] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, Sept. 1992.

[116] J.-P. Sheu, C.-M. Chao, and C.-W. Sun. A clock synchronization algorithm for multi-hop wireless ad hoc networks. In *Proc. of the 24th International Conference on Distributed Computing Systems*, pages 574–581, Tokyo, Japan, Mar. 2004.

[117] K. G. Shin, T.-H. Lin, and Y.-H. Lee. Optimal checkpointing of real-time tasks. *IEEE Transactions on Computers*, C-36(11):1328–1341, Nov. 1987.

[118] M. L. Shooman. *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design.* John Wiley & Sons Inc., New York, 2002.

[119] M. L. Sichitiu and V. Ramadurai. Localization of wireless sensor networks with a mobile beacon. In *Proc. of the 1st International Conference on Mobile Ad-hoc and Sensor Systems*, pages 174–183, Fort Lauderdale, Florida, Oct. 2004.

[120] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *Proc. of 2001 IEEE International Conference on Communications*, pages 472–476, Helsinki, Finland, June 2001.

[121] A. P. Snow, U. Varshney, and A. D. Malloy. Reliability and survivability of wireless and mobile networks. *IEEE Computer*, 33(7):49–55, July 2000.

[122] K.-F. Ssu. *Heterogeneous and Mobile Recovery.* PhD thesis, University of Illinois at Urbana-Champaign, 2000.

[123] H. Takagi. Queuing analysis of polling models. *ACM Computing Surveys*, 20(1):5–28, Jan. 1988.

[124] H. Takagi. Queueing analysis of polling models: Progress in 1990-1994. In J. H. Dshalalow, editor, *Fronties in Queueing: Models and Applications in Science and Engineering*, chapter 5, pages 119–146. CRC Press, 1997.

[125] A. N. Tantawi and M. Ruschitzka. Performance analysis of checkpointing strategies. *ACM Transactions on Computer Systems*, 2(2):123–144, June 1984.

[126] D. Tian and N. D. Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Communications and Mobile Computing*, 3:271–290, May 2003.

[127] D. Tian and N. D. Georganas. Location and calculation-free node-scheduling schemes in large wireless sensor networks. *Ad Hoc Networks*, 2(1):65–85, Jan. 2004.

[128] D. Torrieri. Calculation of node-pair reliability in large networks with unreliable nodes. *IEEE Transactions on Reliability*, 43(3):375–377, Sept. 1994.

[129] D. VanderMeer, A. Datta, K. Dutta, K. Ramamritham, and S. B. Navathe. Mobile user recovery in the context of Internet transactions. *IEEE Transactions on Mobile Computing*, 2(2):132–146, Apr.–June 2003.

[130] U. Varshney, A. P. Snow, and A. D. Malloy. Measuring the reliability and survivability of infrastructure-oriented wireless networks. In *Proc. of the 26th Annual IEEE Conference on Local Computer Networks*, pages 611–618, Tampa, Florida, Nov. 2001.

[131] U. Varshney and R. Vetter. Emerging mobile and wireless networks. *Communications of the ACM*, 43(6):73–81, June 2000.

[132] G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. In *Proc. of IEEE Infocom 2004*, volume 4, pages 2469–2479, Hong Kong, Mar. 2004.

[133] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proc. of the 1st ACM International Conference on Embedded Networked Sensor Systems*, pages 28–39, Los Angeles, California, Nov. 2003.

[134] G. Xing, C. Lu, R. Pless, and J. A. O'Sullivan. Co-Grid: an efficent coverage maintenance protocol for distributed sensor networks. In *Proc. of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 414 – 423, Berkeley, California, Apr. 2004.

[135] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. of the 7th ACM International Conference on Mobile Computing and Networking*, pages 70–84, Rome, Italy, July 2001.

[136] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *Proc. of the 1st ACM International Conference on Embedded Networked Sensor Systems*, pages 51–62, Los Angeles, California, Nov. 2003.

[137] B. Yao and W. K. Fuchs. Proxy-based recovery for applications on wireless hand-held devices. In *Proc. of the 19th Symposium on Reliable Distributed Systems*, pages 2–10, Nurnberg, Germany, Oct. 2000.

[138] B. Yao, K. F. Ssu, and W. K. Fuchs. Message logging in mobile computing. In *Proc. of the 29th International Symposium on Fault-Tolerant Computing*, pages 294–301, Madison, Wisconsin, June 1999.

[139] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. of the 23rd*

International Conference on Distributed Computing Systems, pages 28–37, Providence, Rhode Island, May 2003.

[140] F.-M. Yeh, S.-K. Lu, and S.-Y. Kuo. OBDD-based evaluation of k-terminal network reliability. *IEEE Transactions on Reliability*, 51(4):443–451, Dec. 2002.

[141] Y. B. Yoo and N. Deo. A comparison of algorithms for terminal-pair reliability. *IEEE Transactions on Reliability*, 37(2):210–215, June 1988.

[142] J. W. Young. A first order approximation to the optimum checkpoint interval. *Communications of the ACM*, 16(9):530–531, Sept. 1974.

[143] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. Technical Report UIUCDCS-R-2003-2351, Univ. of Illinois at Urbana Champaign, June 2003.

[144] A. Ziv and J. Bruck. An on-line algorithm for checkpoint placement. *IEEE Transactions on Computers*, 46(9):976–985, Sept. 1997.

[145] Y. Zou and K. Chakrabarty. A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Transactions on Computers*, 54(8):978–991, Aug. 2005.