Analysing Mobile Software Systems: Modeling and Performance Comparison

Fei Gao

M.Sc. Thesis

January 8, 2007

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Analysing Mobile Software Systems:
Modeling and Performance Comparison

by

Fei Gao

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

of

Master of Science

Abstract:
Performance assessment and validation of software systems should be included early on as the cornerstone of software development, regardless of the intended application area of the software system under development. This procedure is extremely important for mobile or nomadic software applications, which are complex and incur substantial performance risks stemming from the conflicting requirements for mobility and connectivity. In this thesis, software systems with mobile components are modeled using the Stochastic Activity Network approach. The resulting models are then used to assess the performance effectiveness of different architectural styles that can be used to build such systems.

Contents

1 Introduction **3**

1.1 Mobility in distributed software systems 4

1.2 Implications for transactional systems 8

1.3 A simple scenario 10

1.4 Thesis organization 12

1.5 Main contributions 13

2 Related Work **14**

3 Modeling Mobile Systems With SANs **20**

3.1 A framework for modeling mobile systems 20

3.2 Modeling physical mobility and system behaviors 23

3.2.1 An application example 23

3.2.2 SAN modeling of the mobile system without handoff 24

3.2.3 SAN modeling of the mobile system with handoff 28

3.3 Modeling code mobility in the mobile software system 32

3.3.1 Modeling Code On Demand paradigm 33

3.3.2 Modeling Remote Evaluation paradigm 34

3.3.3 Modeling Mobile Agent paradigm 35

4 Performance Analysis of Mobile Software Systems **37**

4.1 Adaptation to the system without handoff 37

4.1.1 Plugging COD paradigm into the system without handoff 38
4.1.2 Plugging REV paradigm into the system without handoff 42
4.1.3 Plugging MA paradigm into the system without handoff 43
4.2 Adaptation to the system with handoff 50
4.2.1 Plugging COD paradigm into the system with handoff 50
4.2.2 Plugging the REV paradigm into the system 53
4.2.3 Plugging MA paradigm into the system 54

5 Numerical Results **61**
5.1 Result assessments for the systems without handoff 61
5.2 Result assessments for the systems with handoff 75

6 Conclusions **95**

Chapter 1

Introduction

The rapidly expanding technology of cellular communications, wireless LAN, and satellite services will make it possible for mobile users to access information any time, anywhere. Regardless of their size and computational capabilities, all mobile devices will be equipped with a wireless connection to the internet. The resulting computing environment, often called mobile or nomadic computing, no longer requires the user to stay at a fixed physical and logical position in the network and thus enables almost unrestricted user mobility.

Nomadic computing is concerned with the connection of portable (mobile) computers, and with the migration of software, from a home platform to a remote one. The requirements for mobility and portability create an entire new class of applications. Mobile computing applications require the ability for connected hosts to exchange data via wireless as well as wired connections. The physically mobile hosts form a network whose topology constantly changes. This highly dynamic physical structure supports an even more fluid logical structure composed of application level components that should possess the ability to move among mobile hosts. As a result, the traditional client-server paradigm which is nowadays used to build distributed applications is no longer sufficient for the development of applications in which the components—devices, data, and applications themselves—may freely move in physical and logical space. Mobility is increasingly considered to be an attractive architectural style for distributed applications development since it can lead to

significant improvements in functionality and user comfort [16]. New computing paradigms based on code mobility reconfigure dynamically, at run-time, the binding between the software components of the application and their physical location within a computer network. However, movement of a software component does not come at no cost. This can be described as a simple sending-a-gift-to-a-friend example: is it better for a person to send a gift to someone else, or simply go there and deliver it personally? Hence, the problem boils down to the comparison of the cost of remote interaction with the cost of local interaction and the necessary transfer of data and applications.

Mostly, software models are assessed and validated in light of their functionality. Other, non-functional requirements (such as performance, availability, and scalability, among others), are often deemphasized or even ignored, although they are every bit as important from users' point of view. In particular, non-functional requirements are not discussed in the early phase of software lifecycle, and related decisions are taken later, with significant, and often disastrous effect on the performance and other perceived qualities of the software system in question.

Therefore, performance assessment and validation should be included early on as the cornerstone of software development, regardless of the kind of software application under development. This is even more critical for mobile or nomadic software applications, which are complex and incur substantial performance risks stemming from the conflicting requirements for mobility and connectivity. This thesis proposes an approach in which software systems with mobile components are modeled using Stochastic Activity Network [8], which are then used to assess the performance effectiveness of different architectural styles.

1.1 Mobility in distributed software systems

In mobile computing, users carrying portable devices should have access to data and information services regardless of their physical location or movement behavior. Mobile computing is concerned with the interconnection of mobile devices to stationary hosts, as well as with the migration of software from one platform to another, physically dislocated

one. While the naïve approach simplifies this problem by plain disconnection of the mobile device from one network and reconnection to the other upon arriving at its destination, it is now commonplace to expect that access to computing and communications facilities of different server hosts and to the Internet should be maintained uninterrupted while the user is in transit. This concept is consistent with the main goal of mobile computing, which is to permit users and programs to be as effective as possible, and to maintain all the capabilities in a transparent fashion, regardless of the user's physical location. Mobile distributed systems cover a broad spectrum of software systems, by considering all the forms of mobility (personal, devices, and computational). In this manner, the advent of mobile computing has opened completely new ways for software systems to be conceived and developed. Under this new style of computation, systems can be designed and programmed in more sophisticated ways by taking advantage of the fact that data, code and agents can change location during execution. Mobile systems need to be dynamically configured according to the networked resources that change over time due to the highly dynamic network topology. Thus, we should not only focus on the computational mobility, but also the mobility of devices.

General mobility can be categorized into process migration, physical mobility, and mobile agents [32]. While this classification may be applied to mobile software systems, it is rather coarse to describe all of their different aspects. The rapid growth of Internet has resulted in a number of distributed application patterns: from simple Web pages, through scripts embedded in Web pages and executed on clients and/or servers, to Java applets and small scripts that are downloaded and executed on demand. While the applet code may be mobile itself, it usually runs in an environment that is static, at least during the execution of the applet. Hence, this is just one step in the direction of truly mobile computing where components at all levels are mobile.

From an architectural point of view, mobile distributed systems are most often built on traditional foundations such as client-server and ad-hoc dynamic architectures. However, the topology and structure of the mobile network is dynamic and unstable. Clients are

connecting and disconnecting, moving from one access point to another and organizing spontaneous networks. Mobile computing consists of traveling people, portable wireless computing and communication devices, and computer network infrastructure supporting global connectivity and remote computing. With infrastructure-based networks, base stations handle traffic over mobile terminals in their transmission range, using either licensed or unlicensed frequency bands. In areas in which there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use, wireless mobile users may still be able to communicate through the formation of an ad-hoc network, with which wireless nodes form a dynamic and temporary network according to their respective location, using unlicensed frequency bands. In such a network, each mobile node operates not only as a host but also as a router, forwarding packets for other mobile nodes in the network that may not be within direct wireless transmission range of each other. The above allows deployment of mobile distributed systems that provide mobile users with seamless access to a set of services and content. However, the wireless infrastructure poses specific challenges on the software systems compared to the wired environment. Mobile software systems must in particular cope with the network dynamics and quality of service (QoS) management, as the mobile environment is constrained by limited resources and capabilities of wireless devices and varying bandwidth. However, this aspect is beyond the scope of this thesis, which deals only with device and application mobility in infrastructure-based networks.

There have been many attempts to design a computational infrastructure that can match the characteristics of the communication infrastructure. Most of the approaches are based on well-established models and technology, for example the traditional client-server architecture such as CORBA [36], which integrates remote procedure calls (RPC) with the object-oriented paradigm. Additional capabilities are then added on top of such a model in order to cater to the requirements of mobile computing paradigm. However, extending existing client-server technology is only a partial solution because it addresses just the scaling and connectivity problems without providing new mechanisms and abstractions

that allow the developers to structure the computation in a flexible, customizable and extensible way.

A different approach comes from a promising research area of application mobility, extensively discussed in [16], where the architectural concepts of components, interactions, and sites were introduced as the basis upon which mobile, distributed applications could be developed. Code mobility is not simply process migration, but an application level design paradigm. Subsequent work has identified three mobile code paradigms described in terms of location of, and coordination between the components: Remote Evaluation (REV), Code on Demand (COD), and Mobile Agent (MA) [9, 3] as shown in Figure 1.1.

- The Code on Demand paradigm got widespread with the advent of Java applets. The client requests code from a server, the server returns the binary code, and the client runs this code. In this paradigm, only code is transferred via the network.

- The remote evaluation paradigm also bases on transferring code. The client transfers code to the server, the server executes the code and returns the result to the client; examples include the common object request broker architecture (CORBA) and SOAP.

- Finally, the mobile agent paradigm is a mixture of the previously mentioned paradigms. A mobile agent consists of code, data, and a program state. The key property of it is autonomy: the client sends a mobile agent to the server and the server executes it. After the execution, the mobile agent can autonomously decide to move to another server or to return to the client. In this paradigm, the program code, the data, and the program state are transferred through the network.

As can be seen, these paradigms differ on where code executes and who determines when mobility occurs. These aspects are used to identify and further define the interaction pattern between components, and thus reduce the complexity of designing the software architecture of distributed applications.

In general, mobile code is about moving data and/or code. Code is obtained from a remote system, which is executed on one's local system, as Web applets, executable email attachments, or proxies downloaded as part of a distributed object technology, such as Java RMI or Jini.

In a general sense, mobile agents are also mobile code, since, from the view of the system receiving the mobile agent, *some* code is being downloaded to it. In the mobile agent paradigm, migration of agent code, data and execution state can be described as strong mobility, while the migration of only the agent code and data presents a case of weak mobility.

1.2 Implications for transactional systems

Besides the client-server model, distributed transaction processing is also crucial for the design of a large-scale distributed system. The mobile computing paradigm introduces some new challenges in the area of database systems. Applications of mobile computing will demand various transactional and transaction-like services. Because mobile computers have a nontrivial amount of local storage, transaction processing is at least as complex as in a traditional distributed database system. In mobile computing environments, a mobile host may move to a different cell during the execution of a single transaction. The user may want to continue execution in the new cell or may want to leave his transaction continue in the previous cell. In the former case data related to the host may need to migrate to the new base station. In the later case, the responsibility of the continuation of the transaction must be delegate to some other process. The mobile database is a database that is accessed by one or more mobile devices. Both static and mobile users access data in the mobile database by performing mobile transactions, and different mobile transaction management models have been introduced in [45, 40, 15, 13]. Since users need to be able to work effectively during periods of disconnection, mobile computers will require a substantial degree of autonomy in transaction management. Because of the limitations caused by mobile computing, most mobile transaction models make use of the resources

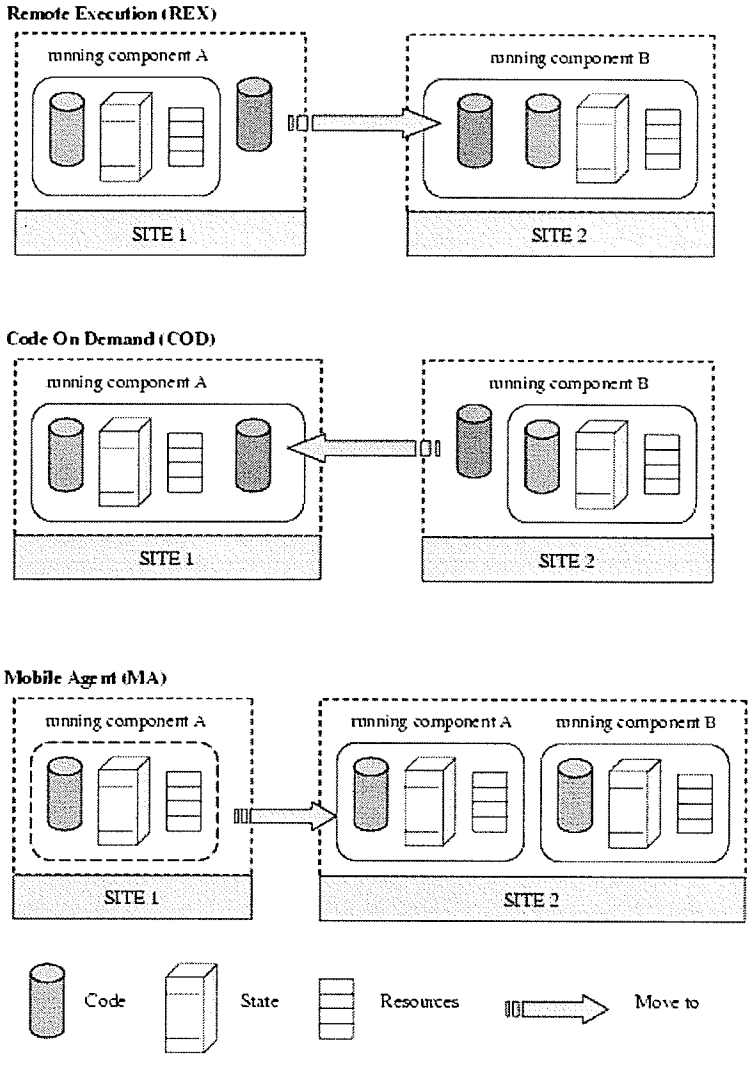


Figure 1.1: Mobile code paradigms

of the fixed network as much as possible. The most general mobile database framework was outlined by [15]. Figure 1.2 shows the model consisting of source systems, data access agents, and mobile transactions. A mobile transaction is "a distributed transaction, where some parts of the computation are executed on mobile and some parts on non-mobile hosts" [39]. Source systems offer information services to the mobile environment. These are stationary systems that register their service offerings with the mobile network. Data in these source systems is accessed by data access agents located on the base stations. When a mobile unit requests a transaction, the data access agent on the base station in the mobile unit's current cell forwards the request to the data source offering the requested service. Therefore, data access agents essentially manage the mobile transaction on behalf of the mobile unit, and are responsible for tracking all the concurrently executing transactions at the base station, and performing logging and checkpointing.

1.3 A simple scenario

Different mobility types presented in mobile software systems can be assessed by considering a simple scenario described below.

Let us consider just one such scenario: a client-server architecture in which the user tries to make a flight reservation and the related hotel booking through a wireless PDA while commuting from her office. The PDA offers an flight reservation application and an related hotel booking application that allow the user to specify the desired price range, the names of cities of departure and destination, the depart time and the return time, or the fare type (cabin) of the flight and the price range, the location, the room type, or the facilities of the hotel in the city of destination. The PDA then sends a query to a flight reservation service, the result of which is a list of flights matching the criteria. The user can select the desired flight, at which point the PDA initiates the following transaction: check availability of seat, debit the user's credit card, credit the airline company's account, enter the reservation into the flight's reservation database, and finally obtain and store a reservation code on the PDA. Physical mobility of the user translates into physical mobility

of the client application, which has to execute handoffs when moving from one wireless cell to another; the data that cannot fit in the PDA memory will have to be kept at the base station and moved together with the user. Since the two reservations must be related, and assuming that the available memory of the PDA is insufficient to hold both the flight reservation application and the hotel booking application, the code for both applications must be mobile as well, so that it can be loaded on demand. Furthermore, user transactions may have to be suspended when the user leaves one wireless cell, and resumed when she is admitted to another cell. In this case, the transaction mobility must be considered. As can be seen, just one scenario as simple as a single user traveling through different wireless cells and accessing data in a remote database, requires almost all possible mobility types, such as physical mobility, code mobility and transaction mobility.

Based on the general mobile database framework that was presented in [22], which consists of source systems, data access agents, and mobile transactions, we consider this scenario as follows. The flight reservation service is a source system that offers information services to the mobile environment, and it is stored on the fixed hosts. Data in the source systems is accessed by data access agents located on the base stations. When the PDA sends the request to reserve a flight seat, the data access agent(DAA) on the base station in the PDA's current cell forwards the request to the data source offering the requested service. The mobile transaction manager (MTM) is executing on each DAA, which is responsible for the management of the mobile transactions Therefore, data access agents essentially manage the mobile transaction on behalf of the PDA. When the mobile host moves from one cell to another, the transactions are handed over to the new base station. Since the PDA may frequently disconnect from the network, the source system can transfer the code and data for execution as a mobile component to the PDA during connection. The disconnection can be the result of low battery power, or the change of the physical location of the PDA.

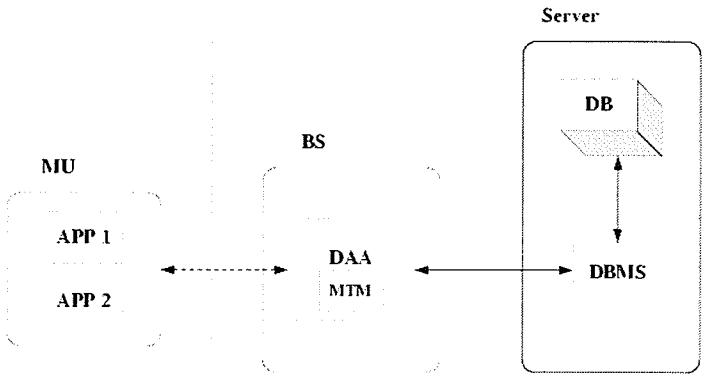


Figure 1.2: Transaction system architecture.

1.4 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 reviews the existing work on the integration of performance analysis into the software development process, especially those related to the mobile software systems where the code mobility is presented, with the comparison of their contributions and drawbacks. Chapter 3 introduces the Stochastic Activity Networks (SANs) [31] used as the framework for modeling mobile systems, describes the UML sequence diagrams of the systems, and indicates the modeling of the behavior of the system and the physical mobility and three code mobility paradigms. Chapter 4 presents the performance evaluation of mobile software systems by means of inserting different code mobility paradigms into the system models to generate the final performance models. Chapter 5 displays the numerical results produced from solving the performance models and compares these results to analyze the performance of different code mobility paradigms. Finally conclusions and future work are discussed in Chapter 6.

1.5 Main contributions

The main contributions of the thesis are as follows.

- It provides a methodology based on the dual foundations of UML modeling notation, in particular the sequence and interaction diagrams, on one side, and SAN modeling framework on the other, that allows for creating performance-oriented models of software systems with mobile components.

- This approach takes into account both physical device mobility as well as the usage of various resources in different phases of execution, and thus allows for more accurate modeling of the impact of different design parameters on the performance of systems under consideration.

- Overall, this approach makes it possible to evaluate the performance of systems in the early stage of the development process, which in turn facilitates the design and construction of systems with better performance such as response time and of better quality.

- It uses this approach and the general methodology to create suitable models for three most widely used paradigms for software systems with mobile components, and to subsequently evaluate and compare their performance. In this process, I have used the Mobius modeling and simulation tool to develop, test, and execute suitable models.

In this manner, I hope that this thesis will present a small contribution to the state-of-the-art of software development for systems with mobile.

Chapter 2

Related Work

Traditional software development methods focus on software correctness, introducing performance issues later in the development process. This style of developing has been often referred to as a fix-it-later approach. In the research community, there has been a growing interest in the subject and several approaches to early software performance predictive analysis have been proposed. Two basic features of these methods are the existence of a performance model suitably coupled with the system software artifacts (software architectures, specifications etc.) and the evaluation of the performance model as the means to obtain software performance results. Since performance is a runtime attribute of a software system, performance analysis requires suitable descriptions of the software runtime behavior. For example, finite state automata and message sequence charts are largely used behavioral models. As far as performance analysis is concerned, we concentrate on model-based approaches that can be applied to any phase of the software life cycle to obtain figures of merit characterizing the quantitative behavior of the system. The most widely used performance models include queueing networks, stochastic Petri nets, stochastic process algebra, and simulation models [26, 23, 2, 5]. Analytical methods and simulation techniques can be used to evaluate performance models in order to get performance indices. These can be classical resource oriented indices such as throughput, utilization, response time, and/or new figures of merit such as power consumption related to mobile software systems.

The approaches that propose a general methodology for software performance focusing on early analysis refer to different specification languages and performance models, and consider different tools and environments for system performance evaluation. Some of the proposed methods are based on the Software Performance Engineering (SPE) methodology which can be defined as the systematic process for planning and evaluating the new system's performance throughout the life cycle of its development [43]. It is a method for constructing software systems to meet responsiveness goals such as response time or throughput. With it, one models software requirements and designs, and evaluates whether predicted performance metrics meet the specified goals. If not, alternatives are proposed and assessed. The SPE methodology was the first comprehensive approach to the integration of performance analysis into the software development process throughout the whole software life cycle. The first approach based on the SPE methodology was proposed in [48], where the SPE methodology was applied to evaluate the performance characteristics of a software architecture specified by using the Unified Modeling Language (UML) diagrams [14], that is, Class Diagrams, Deployment Diagrams, and Sequence Diagrams enriched with ITU Message Sequence Chart (MSC) features [37]. The emphasis is in the construction and analysis of the software execution model, which is considered the target model of the specified software architecture and is obtained from the Sequence Diagrams. Moreover, an extension of the approach developed in [11], called PRIMA-UML, makes use of information from different UML diagrams to incrementally generate a performance model representing the specified system. In PRIMA-UML, software architectures are specified by using Deployment, Sequence, and Use Case diagrams. The software execution model is derived from the Use Case and Sequence diagrams, and the system execution model is derived from the Deployment diagram. Another approach based on CLISSPE (Client/Server Software Performance Evaluation), a language for the software performance engineering of client-server applications [28] was proposed for the design and performance analysis of client-server systems [29]. A CLISSPE specification is composed of a *declaration section* containing clients and client types, servers and server types, database tables and other similar information, a

mapping section allocating clients and servers to networks, assigning transactions to clients, etc., and a *transaction specification section* describing the system behavior. The CLISSPE system provides a compiler which generates the Queueing Network (QN) model, estimates some model parameters, and provides a performance model solver.

Several stochastic extensions of Process Algebras (SPA) [23] have been proposed in order to describe and analyze both functional and performance properties of software specifications within the same framework, such as TIPP (Time Processes and Performability evaluation) [24], EMPA (Extended Markovian Process Algebra) [7], and PEPA (Performance Evaluation Process Algebra) [17], which are all supported by appropriate tools. The advantage of using PEPA, EMPA, or TIPP for software performance is that they allow the integration of functional and nonfunctional aspects and provide a unique reference model for software specification and performance. However, the analysis usually refers to the numerical solution of the underlying Markov chain which can easily lead to numerical problems from the performance evaluation viewpoint, and the software designer is required to be able to specify the software system using process algebras and to associate the appropriate performance parameters to actions. To overcome the last drawback, a method is proposed to derive SPA from UML diagrams [41].

Recently, integration of UML specifications and Petri Nets has been proposed [6, 27], as Generalized Stochastic Petri Nets (GSPN) [1] may be derived from different UML diagrams. In [27], a GSPN is derived from Use Cases diagrams and combined Collaboration and Statecharts diagrams, and [6] proposes a systematic translation of Statecharts and Sequence Diagrams into GSPN, with the two types of diagrams being translated into two separate labeled GSPNs.

All of the approaches for the performance evaluation described above were developed with non-mobile software systems in mind. In fact, there are few proposals specifically tailored for performance evaluation and modeling of mobile software systems [18, 8, 4, 25, 21, 20, 19, 30]. Among these approaches, [25] and [30] typically focus on the performance evaluation of the Mobile Agent (MA) paradigm. The former compares the performance of

the mobile agent paradigm with the client-server paradigm by performing the evaluation in the Java environment using RMI [47], the Aglets mobile agents platform [44], and a minimal mobile agents platform developed for the purpose of the simulation. The latter models the system using UML notation and extends UML to include performance-related aspects. The corresponding formal Petri Net [38] model can be obtained from the UML representation through the use case diagrams, the sequence diagrams, and the state transition diagrams. Then, the performance results may be computed and assessed. My proposed approach is similar to the methodology described in [30], except that my proposed approach only makes use of the UML sequence diagrams to model mobile systems. The approach in this thesis uses the Mobius tool [12] to simulate and solve the SAN models, and thus covers both physical mobility and three code mobility paradigms.

Most of the methodologies for modeling and performance analysis of mobile software systems adopt and extend the UML notations for system modeling and software architecture description, but generate different performance models. PRIMAmob-UML [20] provides a performance analysis model that presents the definition of a general methodology that can be used as a support for the designer during the early phases of the development of a specific software application, providing insights about the consequences of architectural choices that can be taken during those phases, and an extension of UML (Unified Modeling Language) to model the mobility based paradigms in software architectures. This methodology is an extension of PRIMA-UML methodology [11] in order to cope with the case of mobile software architectures by enhancing its UML description to model the mobility-based paradigms and allows the designer to get insights about the impact of the possibly adopted code mobility paradigms on the quality attributes of software systems. In PRIMAmob-UML, component mobility is modeled using both collaboration and sequence diagrams, with a separation of concerns between them [20]. The approach generates the corresponding software and system execution models allowing the designer to evaluate the convenience of introducing logical mobility with respect to communication and computation costs. The authors define extensions of EG (Execution Graphs) and EQN (Extended Queueing Net-

work)to model the uncertainty about the possible adoption of code mobility. Another UML based approach presented in [19] introduces an UML extension to modeling the possible adoption of mobility-based paradigms in the software architecture of an application and generates a stochastic performance model which is a Markov Reward Process (MRP) or a Markov Decision Process (MDP) [42] to help making the decision of introducing different code mobility paradigms to the software system. This approach makes use of both UML collaboration and sequence diagrams and presents an extension to the collaboration diagrams to model the possibly adopted code mobility paradigms, which has the same idea as the PRIMAmob-UML [20], however, they generate different types of performance models.

Grassi *et al.* have also developed an UML based modeling framework [21] to evaluate the performance effectiveness of different code mobility paradigms to the mobile system, and this framework differs from other approaches by plugging the code mobility paradigms into pre-existing architecture models in different physical mobility scenarios prior to the actual system implementation. In this framework, the performance analysis is performed by applying graph analysis techniques to the augmented UML Activity Diagrams (ADs). Moreover, in [4], Balsamo *et al.* introduce a unified UML notation for high-level description and performance modeling of mobile systems which supports the quantitative information used to build a process-oriented simulation model of the system, and the simulation model is executed to evaluate the performance of the mobile system with the results returned back to the original UML diagrams. The simulation model consists of three types of simulation processes derived from the UML diagrams which are Workload, Scenarios and Resource processes [4]. A prototype tool is presented for translating annotated UML models into simulation programs. This framework focus on the integration of performance modeling analysis with software system specification for mobile systems. These two methodologies [21, 4] consider both physical device mobility and code mobility.

Besides those UML based approaches, [18] introduces a methodology that helps the designer to perform the performance evaluation of client-server style and code mobility in applications at the software architecture level. This methodology includes adopting an

architecture description language (ADL) which is a modification of the COMMUNITY language [46] to describe the software architecture of the application, deriving a labeled transition system (LTS) [18] from the software architecture description to represent the application dynamics, and deriving a Markov Decisional Process (MDP) [42] from the LTS as the performance model that can be solved to evaluate the performance measures. The main drawbacks of the work include no consideration of physical mobility and no distinct separation of three code mobility paradigms. This approach is further developed and completed in [10].

Finally, some authors only aim at performance model generation and analysis [8]. This framework allows the early performance evaluation of mobility-based software systems by using Stochastic Activity Networks (SANs) [31], and it separates the performance model into three sub-models that are the behavior sub-model, the integrity sub-model and the mobility sub-model. In this framework, the behavior sub-model represents the dynamic behavior of the system, and the mobility sub-model represents the code mobility paradigms by exploiting the UML collaboration diagrams. Finally, the performance model is solved by using the Mobius simulation solver [12]. Even though my proposed methodology also makes use of SANs as the modeling tool and the Mobius to solve the models, the framework in [8] refers only to the Mobile Agent paradigm and does not cover the physical mobility in the system, comparing with my proposed approach.

Chapter 3

Modeling Mobile Systems With SANs

There are different techniques for modeling and analysis of communication systems and networks. Petri Nets (PNs) [38] and their extensions are formalisms for modeling and analysis of concurrent and distributed real time systems. Hence, a framework based on a variation of Petri Nets is adopted in this work for modeling mobile systems, which will be described in this chapter.

3.1 A framework for modeling mobile systems

A general framework based on a variation of Petri Nets [35] known as the Stochastic Activity Networks, or SANs [31] is used to model mobile systems. In this framework, an activity network - a set of activities, their durations, and precedence constraints - is represented by a directed acyclic graph (DAG) in which each node corresponds to a specific activity. Activity network models have been used in many fields, such as project management and parallel processing systems.

Activity networks

We consider a set of n activities, labeled 1,...,n, which are carried out under some given precedence constraints among them, i.e., each activity starts only when certain others have finished. We represent the precedence constraints by a directed acyclic graph (DAG) $G = (N, A)$, called the *precedence graph*, with nodes $N = \{1, ..., n\}$ and arcs $A \subseteq \{(i, j) : i, j = 1, ..., n\}$. Each arc in the precedence graph represents a precedence constraint: $(i, j) \in A$ means that activity j cannot start until activity i ends. For each activity $k \in N$, we denote the set of its *predecessor nodes* (called *fan-in* in IC design) as $\mathbf{Pred}(k)$, and the set of its *successor nodes* (called *fan-out* in IC design) as $\mathbf{Succ}(k)$:

$$\mathbf{Pred}(k) = \{i \in N : (i, k) \in A\}, \mathbf{Succ}(k) = \{j \in N : (k, j) \in A\}$$

Input (also called *source* or *starting*) nodes are those with no predecessors, and output (or *sink* or *ending*) nodes are those with no successors. We let $d_i > 0$ denote the duration (or processing time or delay) of activity i. The pair (G, d), i.e., a DAG along with a duration for each node, is referred to as an *activity network*.

A path on the DAG G represents a sequence of activities that must occur in order. The *duration* or *delay* of a path is the sum of the durations of the activities on the path. For each activity we define its *completion time* as the maximum of the durations of all paths finishing at the activity. The maximum of the durations of all paths, which is also the maximum of the completion times for all activities, is called the *makespan* of the activity network. The makespan is also called the total delay, total processing time, or total duration. The makespan is exactly the total time for all activities to complete, when the starting activities start at $t = 0$, and each acitivity starts when its predecessors have finished. Any path with the maximum of the durations of all paths is said to be *critical*, and an activity is said to be critical if it is on a critical path.

Stochastic activity networks

Stochastic activity networks (SANs) which have been introduced in 1984, are a stochastic extension of PNs. SANs are more powerful and flexible than most other stochastic exten-

sions of Petri Nets including stochastic Petri Nets (SPNs) [33] and generalized stochastic Petri Nets (GSPNs) [1]. A stochastic activity network (SAN) is an activity network with random activity durations. A SAN is described by the pair (G,$\mathbf{D}$), where the (DAG) $G = (N, A)$ is the precedence graph, and ($\mathbf{D} = \mathbf{D}_1, ..., \mathbf{D}_n$) is the vector of random activity durations. In a SAN, all quantities that depend on the activity durations are random variables. In particular, the delay of any path, completion time at any node, and the makespan are all random variables [31].

In a new definition of SANs [34], there are five primitives: place, timed activity, instantaneous activity, input gate and output gate. Activities represent actions of the modeled system, and they are of two types: timed and instantaneous. Timed activities have durations that impact the performance of the modeled system, and they are represented graphically in DAG as thick vertical lines. Instantaneous activities represent actions that complete immediately when enabled in the system, and they are represented graphically in DAG as thin vertical lines. Places represent the state of the modeled system, and they are represented graphically as circles. Each place contains number of tokens, which represents the marking of the place. The tokens in a place are homogeneous, in that only the number of tokens in a place is known. The meaning of the marking of a place is arbitrary. The set of all place markings represents the marking of the stochastic activity network. Furthermore, input gates control the enabling of activities and define the marking changes that will occur when activities complete, and they are represented graphically as triangles. Output gates define the marking changes that will occur when activities complete, and an output gate is represented graphically as a triangle with its flat side connected to an activity or a case. An activity may have different cases describing different possibilities occurring in an application. The only difference from an input gate is that an output gate is associated with a single case. For timed activities, definition of general probability distribution function is possible. For input and output gates, general functions, written in languages like C could be defined. To be able to fire in a marking, the enabling predicates of all input gates of an activity must be true.

In this modeling framework, we will first describe mobile system behavior by using UML Sequence Diagrams (SDs), and then model the system by SANs which are derived from corresponding SDs. In mobile computing environment, a mobile unit may stay in one cell or move to another cell while working. If the mobile unit stays in one cell, the mobile system does not need to deal with the handoff with different base stations. When the mobile unit moves to another cell during its working and connecting with the base station, the system has to manage all the handoff between the original base station and the new one. Hence, two cases have to be taken into account in system modeling. Furthermore, code mobility paradigms also will be modeled by SANs and further plugged into pre-existing system models to generates the final performance models.

3.2 Modeling physical mobility and system behaviors

This section indicates how to model the physical mobility and the behavior of the mobile system by graphical stochastic activity networks. A modeling tool called Mobius [12] is exploited for such kind of modeling because it introduces techniques for constructing hierarchical and composed SAN models.

3.2.1 An application example

In this section, an application example is presented for system modeling. The application is the simplified version of the scenario introduced in Chapter 1.

The flight reservation application provides the ability for the user to query for the desired flights from the database located on the server which will return a list of flights matching the criteria. Then the user selects the most wanted one and saves the reservation into the flight database, finally a reservation code will be sent and stored on the mobile unit through which the user makes the flight reservation. In this case, the mobile unit starts the flight reservation application as soon as the user begins to query for the flights. The base station forwards the query message to the server and returns the query results to

the mobile unit. Afterward, the application will initiates a serial of transactions containing some write operations that the application updates the flight reservation database on the server and some read operations that the application reads and stores a reservation code or other information as needed.

3.2.2 SAN modeling of the mobile system without handoff

The user starts the flight reservation application on the mobile unit to query for the suitable flights, and initiates some transactions over the most desired flight. The mobile unit first sends the query message to the server, forwarded by the base station. The server then processes the query and sends the results to the mobile unit through the base station. Whereafter, the mobile unit initiates some transactions. After initiated, the transactions are forwarded to the server through the base station which is responsible for the current cell for accessing the data on the server. We consider the base station manages all of the transactions and keeps a transaction log file. The server processes them after it has received all of the transactions, and these transactions may be committed or aborted. Furthermore, the assumption that only the information of commission of the transactions has to be written to the log file on the base station is made for modeling. Only if all the transactions are committed, the base station will then return a confirmation message containing the reservation code to the mobile unit. In this case, the mobile unit is assumed to stay in one cell while it is working. The system behavior is illustrated by the UML sequence diagram, shown as Figure 3.1. The whole system is modeled by two SAN submodels based on the communications between different components of the system. At last, the individual submodels are joined together to represent the whole system.

Figure 3.2 shows the SAN submodel of the communications between the mobile unit and the base station dominating the cell where the mobile unit is. In this submodel, the places *start,wait_flights,wait_results* are the states of the mobile unit, and the places *flights_sent,end* are the states of the base station. The places *s_flights_sent* and *log_comp* are the shared states of the server appearing in both submodels. These states all present

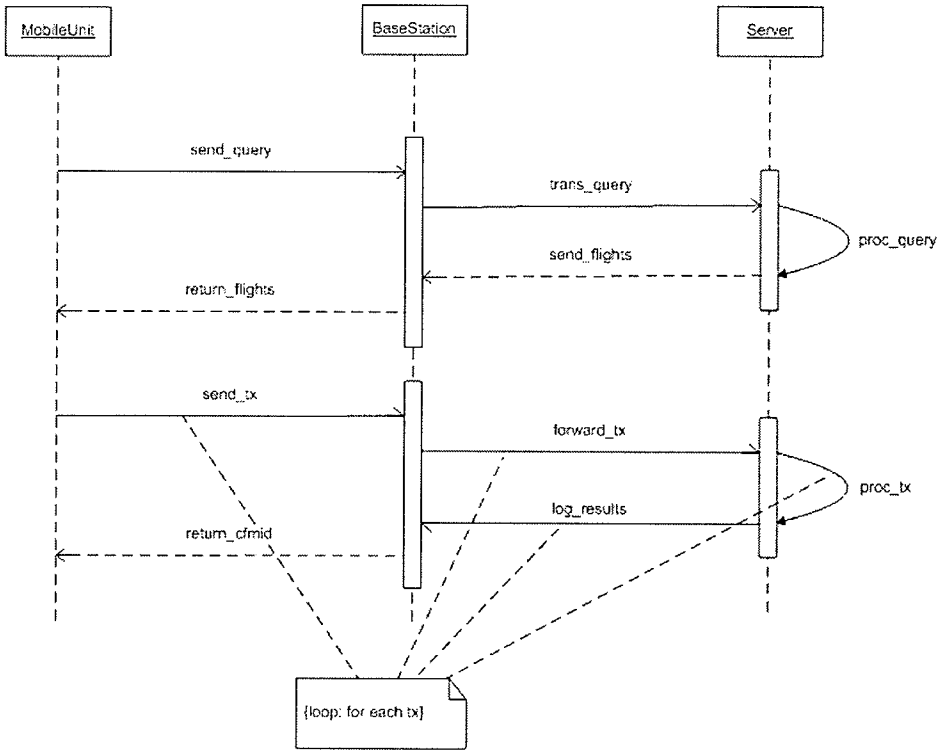


Figure 3.1: The sequence diagram of the system without handoff

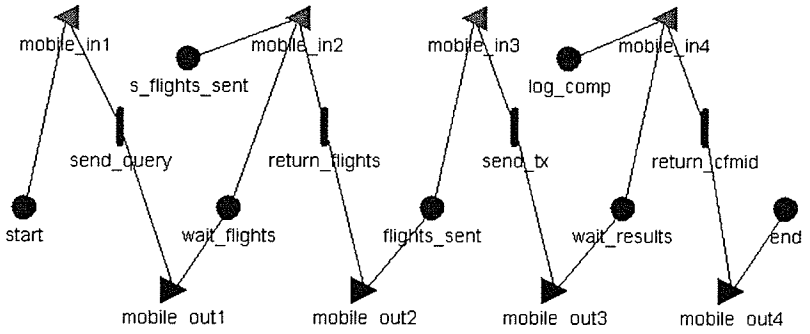


Figure 3.2: SAN submodel of wireless communications in the system without handoff

the system components at the moment when they finish sending a message. If the marking of the sender's state has changed, it means the receiver successfully gets the message. We only define the marking of the original state of the mobile unit to be *mu_msgnum* that denotes the number of the messages to be sent by the mobile unit. Then the original marking of the base station should be *mu_msgnum+2*, and that of the server should be *mu_msgnum*. Because the mobile unit sends the query message and some transactions to the base station, the total number of the messages should be the number of the transactions plus 1. The timed activities *send_query* and *send_tx* represents the actions of the mobile unit when it sends the query message and the transaction messages, and the timed activities *return_flights* and *return_cfmid* represents the actions of the base station when it returns the query result and the confirmation message. The input gate *mobile_in4* enables the activity *return_cfmid* only when all the transactions on the server have been committed and the results have been logged to the log file, denoted by the places *log_comp*.

Figure 3.3 indicates the SAN submodel of the communications between the base station and the server. The places *bs_wait_flights* and *bs_wait_results* represents the states of the base station, and the places *flights_out,s_flights_sent,tx_aborted,tx_commited* and *log_comp* represents the states of the server. The mobile unit's states *wait_flights* and *wait_results* are shared states appearing in both submodels. The timed activities *trans_query* and *forward_tx*

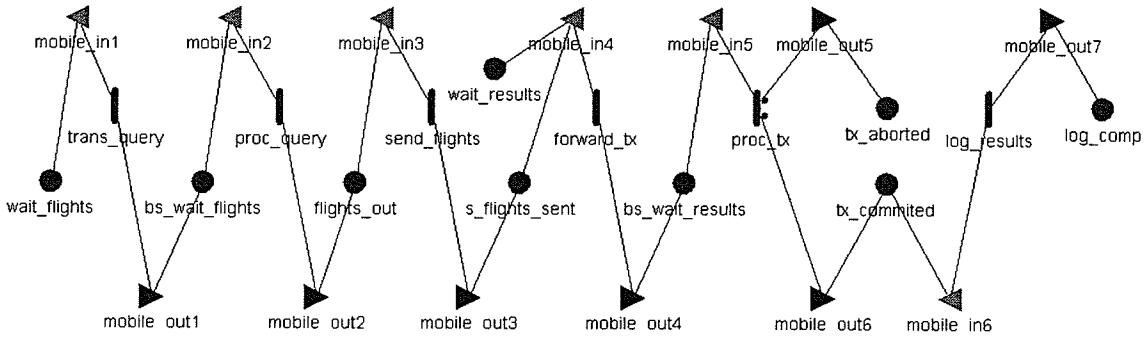


Figure 3.3: SAN submodel of wired communications in the system without handoff

represent the actions of the base station that it transfers the query message and forwards the transaction messages to the server. The server processes the query and sends the query result containing a list of flights to the base station, which are denoted by the timed activities *proc_query* and *send_flights*. Moreover, the server processes the transactions after it has received them, and logs the results into the log file located on the base station. Since each transaction should be either committed or aborted, and the server logs the results only when all the transactions are committed, the server has to log the times as the number of the transactions have been processed, and these actions are represented by the SAN timed activities *proc_tx* and *log_results*. The activity *proc_tx* has two cases according whether all the transactions are committed. Only if all of the transactions have been committed successfully, the server is in the state *tx_commited*; otherwise, it will be in the state *tx_aborted*.

Figure 3.4 shows the replicate/join composed model of the system. The formalism enables the modeler to define a composed model in the form of a tree, in which each leaf node is a predefined submodel node, and each non-leaf node is classified as either a *Join* node or a *Replicate* node. The root of the tree represents the complete composed model. Two submodels, *wireless* and *wired*, make up the composed model of the mobile system without handoff. The shared states in the submodels are defined in the composed model.

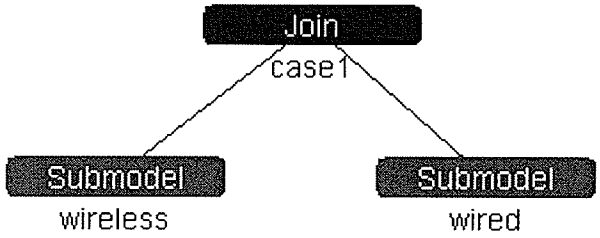


Figure 3.4: SAN composed model of the system without handoff

3.2.3 SAN modeling of the mobile system with handoff

The second case for the mobile system is that the mobile unit moves to another cell which is dominated by another base station during its working time. In this case, mobile applications have to execute handoff when moving from one wireless cell to another. Furthermore, mobile transactions may have to be suspended when the mobile unit leaves one wireless cell, and resumed when it is admitted to another cell. In this case, the mobile unit is assumed to move after all the transactions have been forwarded to the server. It has to inform the original base station the new location of the mobile unit, and the old base station sends a handoff request message to the new base station that will check its capacity for accommodating all the handoff information including transaction id and some resources such as code or data. The new base station returns to the old one the message of admission, and if admitted, the old base station suspends the transactions on the server and then transfers all the handoff information to the new one which will resume the transactions. Afterward, the server processes the transactions and logs the results to the log file. The behavior of the system components is illustrated by UML sequence diagram, as shown in Figure 3.5.

Figure 3.6 indicates the SAN submodel of the wireless communications between the mobile unit and two base stations dominating two cells across which the mobile unit moves. Since the mobile unit migrates after the original base station forwards all the transactions to the server, the query process modeling is the same as that in the system without handoff. The activity *move* represents the event that the mobile unit changes its location, and

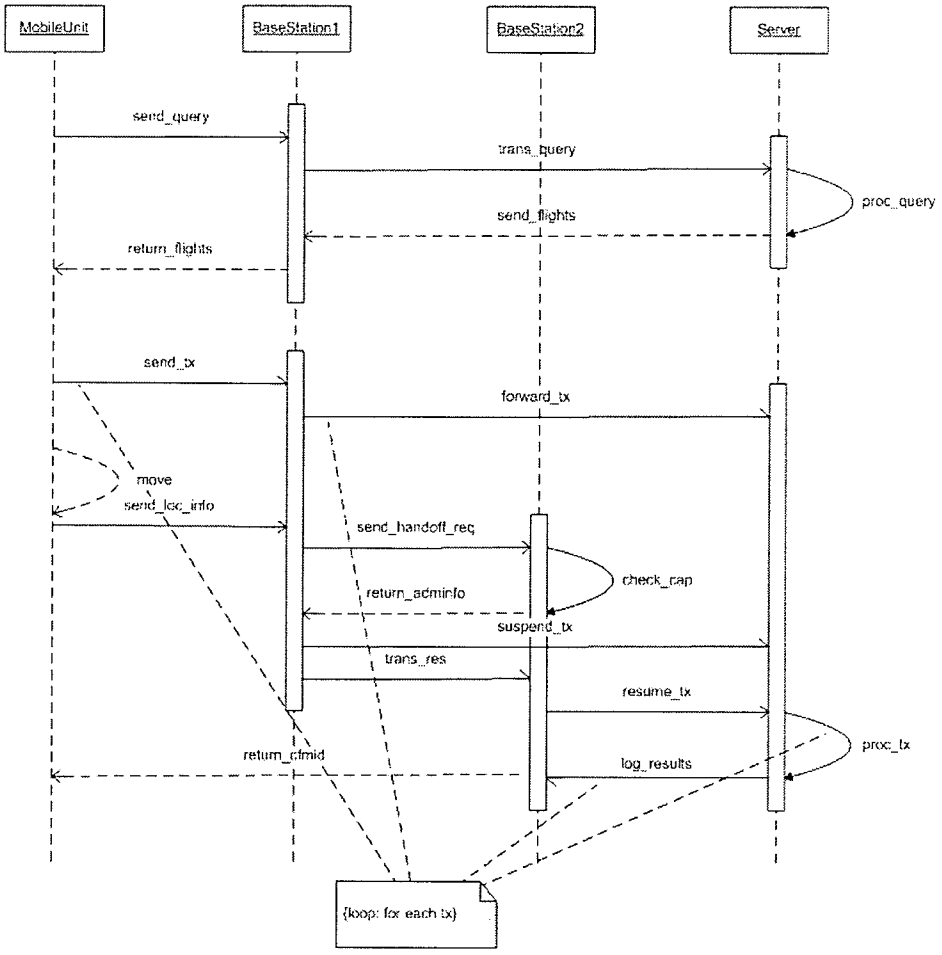


Figure 3.5: The sequence diagram for the system with handoff

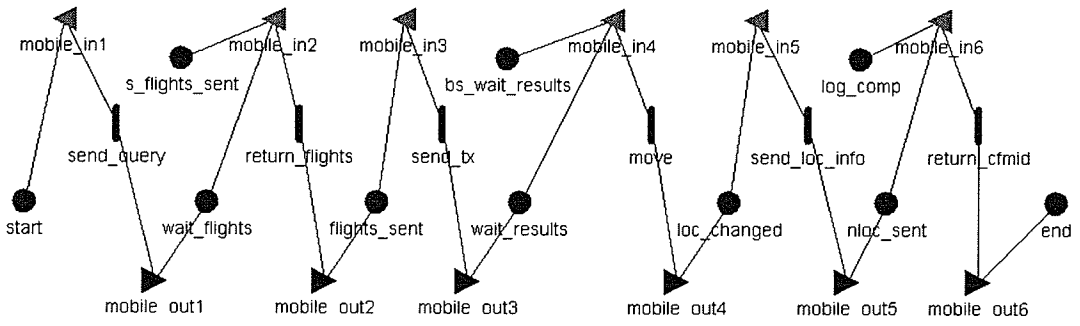


Figure 3.6: SAN submodel of wireless communications in the system with handoff

this activity occurs after the transactions have been delivered to the server. For modeling purpose, the migration of the mobile unit is defined as timed activity with the time distribution rate of 0.001 because the effect of the migration time on the performance measure variable is small enough to be safely ignored. As soon as the mobile unit migrates to another cell, the notification of its new location will be sent to the original base station, denoted by the timed activity *send_loc_info*. Furthermore, in this submodel, the timed activity *return_cfmid* represents the action of the new base station dominating the current cell of the mobile unit when it returns the confirmation message to the mobile unit. Besides the shared states the same as those in the SAN submodel of the other case, the place *bs_wait_results* also denotes the shared state of the original base station after it forwards the transactions to the server. The places *loc_changed* and *nloc_sent* represent the states of the mobile unit after it moves to the new cell and notifies the new location to the original base station respectively.

Figure 3.7 indicates the SAN submodel of the wired communications between two base stations and the server in the system with handoff. In the system, the original base station begins to send the handoff request to the new base station after it forwards all the transactions to the server and receives the notification of the new location of the mobile unit, and this event is denoted by the timed activity *send_handoff_req*. The activities *check_cap*

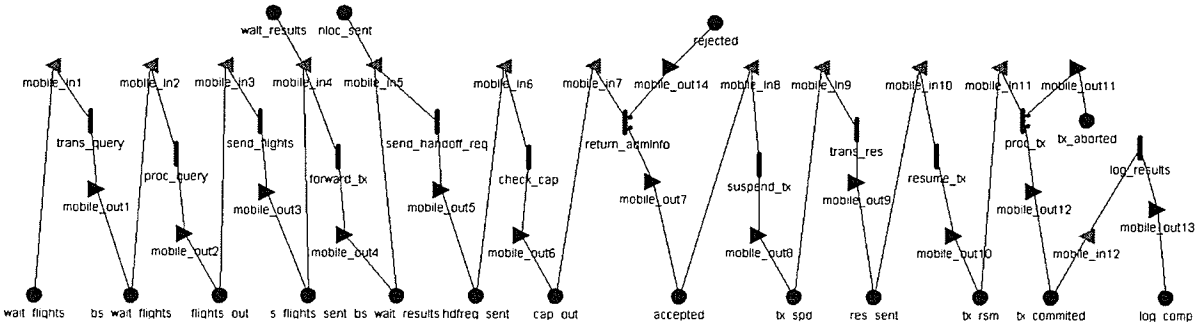


Figure 3.7: SAN submodel of wired communications in the system with handoff

and *return_adminfo* represent the events that the new base station checks its capacity for accommodating all the handoff information and returns the admission information. The activity *return_adminfo* has two cases according to whether the new base station accepts or rejects the handoff request. If the new base station has rejected the handoff request, it is in the state *rejected*; otherwise, it is in the state *accepted*. Only when the handoff request is accepted, the original base station can suspend the transactions on the server, denoted by the timed activity *suspend_tx*. Afterward, it transfers the resources which may be needed by the mobile application and transactions to the new one, represented by the timed activity *trans_res*. As soon as the new base station receives the resources, it resumes the transactions on the server, represented by the timed activity *resume_tx*. Finally, the server has to log the results of processing the transactions to the log file on the new base station which then will return the confirmation message to the mobile unit. The place *nloc_sent* denotes the state of the mobile unit which is shared in two submodels. Figure 3.8 indicates the replicate/join composed model of the system, and all of the shared states are defined in this model. Two submodels *wireless* and *wired* are joined together to form the whole model of the system.

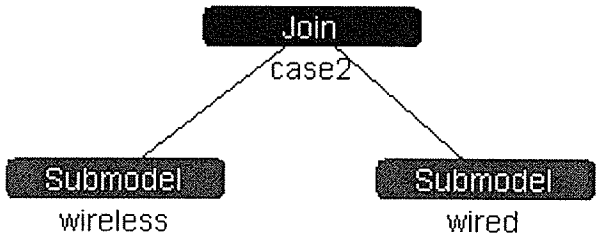


Figure 3.8: SAN composed model of the system with handoff

3.3 Modeling code mobility in the mobile software system

This section presents how to model the three code mobility paradigms by SAN models. In the mobile system described previously, the memory of the mobile unit may not be sufficient to hold many applications; hence, the code for the applications must be mobile so that it can be loaded on demand for execution. In fact, this scenario can be introduced contrarily. The mobile unit sends the code to the base station which also keeps the data for the mobile applications for execution, and the base station returns the results back to the mobile unit. Both of the Code On Demand paradigm (COD) and the Remote Evaluation paradigm REV are about transferring the code or the service; therefore, three components have to be concerned: the mobile unit, the base station and the code constituent itself. In addition, the mobile unit can send the code and data for execution as a mobile component to the server during connection since the mobile unit may frequently disconnect to the network, and this scenario presents the Mobile Agent paradigm (MA). In respect that this paradigm is concerning moving the whole software component in the system, the mobile unit, the base station and the mobile agent itself needs to be modeled. In the modeling, the states of code fragments and mobile agents are expressed by their locations.

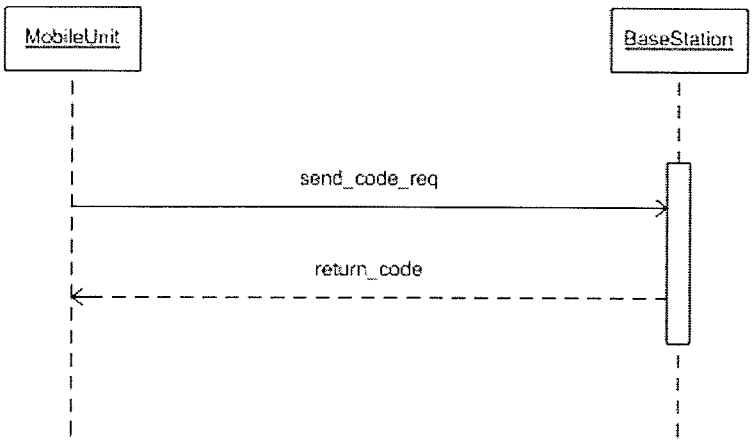


Figure 3.9: The sequence diagram of the COD paradigm

3.3.1 Modeling Code On Demand paradigm

In this paradigm, the mobile unit sends the request for code to the base station which will return the code for execution after receiving the request. The UML sequence diagram for this process is shown in Figure 3.9. We assume that the mobile application automatically loads the code as long as it is started. Hence, the mobile unit will send the query message immediately after it gets the code.

Figure 3.10 shows the SAN model of the COD paradigm. The place *start* represents the original state of the mobile unit which has the marking of 1. The timed activity *send_code_req* forces the state of the mobile unit to be *wait_code* which means the mobile unit has sent the request for code to the base station and it is waiting for the code. The location of the code implies its state. The code is originally located on the base station; after being transferred, its location changes to the mobile unit. Thus, the places *loc_bs* and *loc_mu* represents the two different locations of the code. The timed activity *return_code* represents the behavior of the base station that it returns the code to the mobile unit, enabled by the input gate *mobile_in2* when the mobile unit is in the state *wait_code* and the code is in the state *loc_bs*. The place *code_sent* denotes the state of the base station after the activity *return_code* completes.

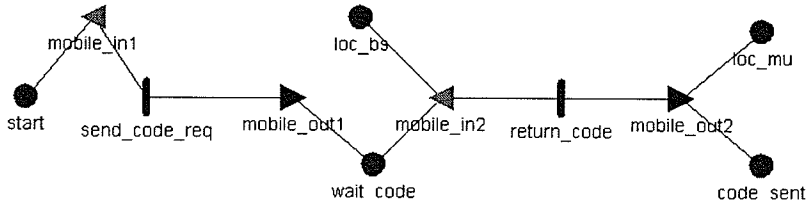


Figure 3.10: SAN model of the COD paradigm

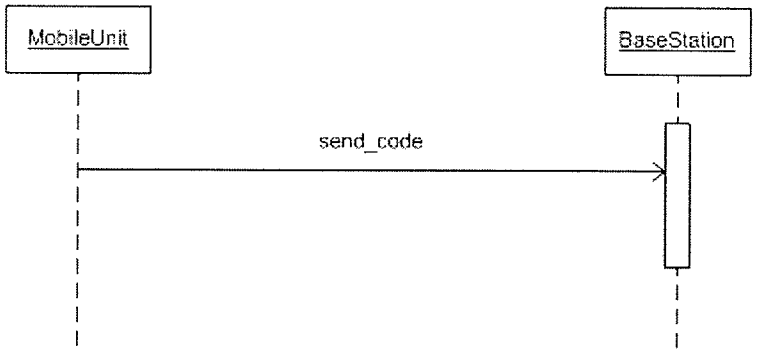


Figure 3.11: The sequence diagram of the REV paradigm

3.3.2 Modeling Remote Evaluation paradigm

In Remote Evaluation paradigm, the server component offers its computational power and resources, but does not provide any application specific service. As in our mobile software system, the mobile unit sends the code to the base station which has other resources for running the application. The mobile unit is assumed to send the code as soon as the application is started. Once the base station has received the code, it sends the query message to the server, but for the modeling of the REV paradigm, only the messages for transfering the code is concerned. The UML sequence diagram for this paradigm is indicated in Figure 3.11.

Figure 3.12 demonstrates the SAN model of the REV paradigm. The timed activities *send_code* represents the mobile unit's behavior of sending the code to the base station,

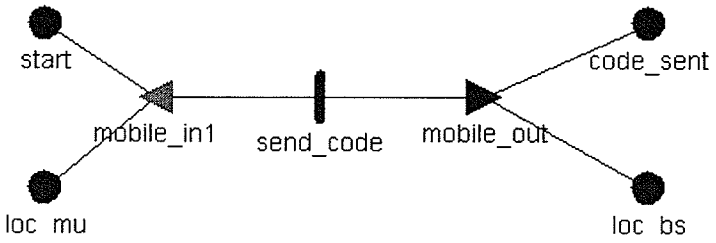


Figure 3.12: SAN model of the REV paradigm

enabling the location change of the code constituent from the mobile unit to the base station. The place *start* represents the original state of the mobile unit, and the place *code_sent* represents the state of the mobile unit after the code has been sent to the base station.

3.3.3 Modeling Mobile Agent paradigm

In Mobile Agent paradigm, the mobile agent that contains the data, the code or the states moves across different system components. When the application is started, the mobile unit sends the mobile agent to the base station which will then forward it to the server. The mobile agent executes on the server and produces the query results which will be returned to the mobile unit. The same as the modeling of the REV paradigm, only the process of transferring the mobile agent will be modeled. The UML sequence diagram of the MA paradigm is displayed in Figure 3.13.

Figure 3.14 demonstrates the SAN model of the MA paradigm. While the mobile agent is executing on the mobile unit, it has to be transferred to the server where it will resume the execution. The input gate *mobile_in1* enables the timed activity *send_ma* from two states of the mobile agent *executing* and *loc_mu* representing the original location of the mobile agent. The output gate *mobile_out1* defines that the state of the mobile agent changes to *loc_bs* which mean the mobile agent is on the base station. The place *ma_sent* denotes

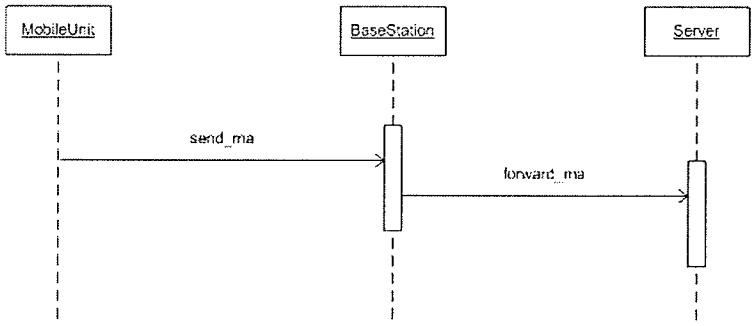


Figure 3.13: The sequence diagram of the MA paradigm

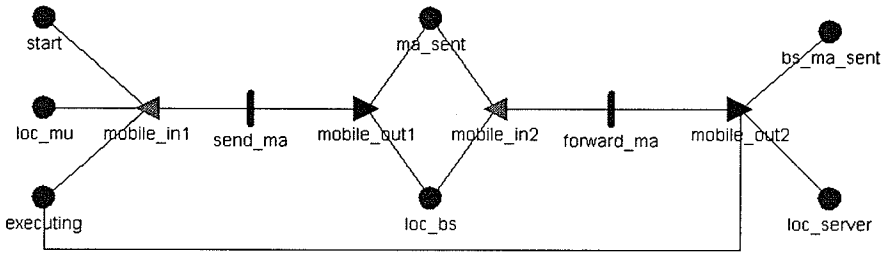


Figure 3.14: SAN model of the MA paradigm

the state of the mobile unit after it has sent the mobile agent. The activity *forward_ma* represents that the base station forwards the mobile agent to the server, and the output gate*mobile_out2* defines that the states of the mobile agent turn to *loc_server* and *executing* which mean that the mobile agent is resumed when it migrates to the server. The place *bs_ma_sent* indicates the state of the base station after it has forwarded the mobile agent to the server.

Chapter 4

Performance Analysis of Mobile Software Systems

As we have existing basic system models, we can plug three mobile code paradigms into those models and compare their performance to get insights about how introducing the code mobility into the wireless communication environments affects the performance of the whole system, which helps the software designer to exam different strategies and figure out the best solution. The method proposed to solve the problem is to insert the SAN models of three mobile code paradigms into two pre-existing mobile system models and assess the performance, finally obtain the simulation results by the Mobius tool.

4.1 Adaptation to the system without handoff

As indicated above, two cases that the mobile unit stays in one cell and it moves to another cell during working time should be considered. Adaptation of three code mobility paradigms to the mobile system without handoff will fist be presented. In such a system, the mobile unit only communicates with one base station on which some data and code may be kept.

4.1.1 Plugging COD paradigm into the system without handoff

In the Code On Demand paradigm, the code for the flight reservation application is located on the base station, and can be loaded by the mobile unit which holds the resources and computational component for the application. The mobile unit should launch the application immediately when the user begins the query, which means the application should start to load the code to perform the task at the same time. As soon as the application has the code, it begins to send the query message to the base station which then forwards the message to the server and returns the query results back to the mobile unit after processing the query by the server. Afterward, the mobile unit sends the transactions messages initiated by the application to the server through the base station. When all the transactions are committed, the server must log the results to the log file located on the base station which then sends a confirmation message including a reservation code to the mobile unit which will end the application finally. The events taking place in the system are represented by the UML Sequence Diagram shown in Figure 4.1.

The whole SAN system model is divided into two SAN submodels which represent the communications through the wireless connection and those through the wired connection respectively, illustrated in Figure 4.2 and Figure 4.3. Two submodels will be joined together as a SAN composed model of the whole system shown in Figure 4.4. The actions in the SAN models correspond to the messages in the UML sequence diagram.

In Figure 4.2, the place *start* represents the state of the mobile unit right at the moment when it starts the application, and the activity *send_code_req* represents that the mobile unit sends the request for the code to the base station. The mobile unit will then wait for the code after it sends the request message, which is represented by the place *wait_code*. At this time, the code is located on the base station, and the place *loc_bs* denotes the original location of the code. When the mobile unit is in the *wait_code* state and the marking equals to the number of subtracting 1 from the original message number of the mobile unit and the code is in the *loc_bs* state, the base station sends the code to the mobile unit, which is represented by the activity *return_code*. The place *code_sent* denotes the state of the

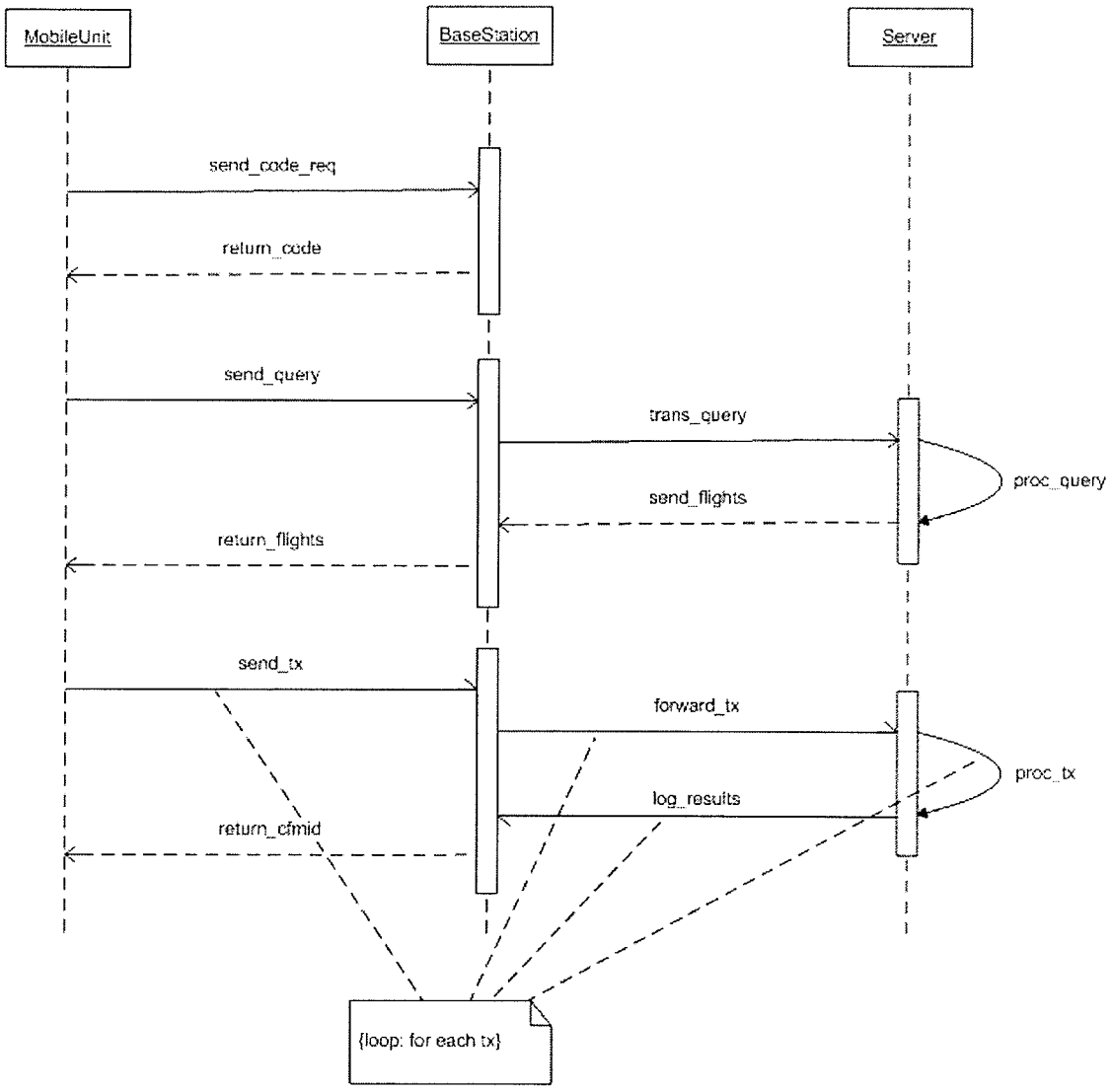


Figure 4.1: Sequence diagram of plugging COD paradigm into the system without handoff

base station after it returns the code. If the mobile unit receives the code successfully, the location of the code should be denoted by the place *loc_mu*. Until now, the code loading is complete, and the application begins to send the query message to the base station which will then forward the message to the server. The mobile unit is in the state *wait_flights* after the query message are sent to the base stations.

When the base station receives the query message, it transfers the message to the server, denoted by the timed activity*trans_query*, and the place *bs_wait_flights* denotes the state of the base station after the action completes. Afterwards, the server processes the query and gets the result, which are represented by the timed activity *proc_query* and the place *flights_out* respectively. The server sends the outputting result to the base station which then returns it to the mobile unit. These actions are divided into two parts which appear in two submodels, *wait_flights* and *s_flights_sent* are shared states in two submodels. When the mobile unit finishes the query, it begins to send the transactions to the server through the base station, and this process is expressed by the actions *send_tx* and *forward_tx*, also appearing in two submodels. The server deals with the transactions, and the activity *proc_tx* has two cases resting with the probability of the transaction to be committed or aborted. The server is in the state *tx_committed* only when all the transactions are committed and it has to log the result to the log file located on the base station. *wait_results* and *log_comp* are the shared states in two submodels. We assume the base station has to receive all the transactions until it forwards them to the server.

When the SAN models of the system are ready, the reward functions are defined to measure information about the system being modeled. Mobius provides performance variables for the measurement. The performance variables needed to be assessed include *download_time*, *query_time* and *tx_execution_time*. The reward function for each performance variable is indicated as below in Figure 4.5. The performance variable *download_time* represents the time of the mobile unit to load the code from the base station, *query_time* represents the time of completing the query for flights, and *tx_execution_time* represents the time of executing all the transactions, starting from sending the transaction messages to

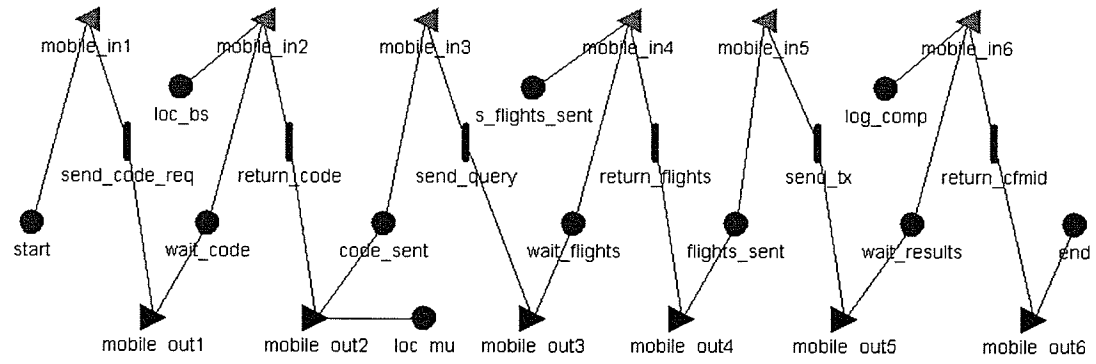


Figure 4.2: SAN submodel of wireless communications in the system

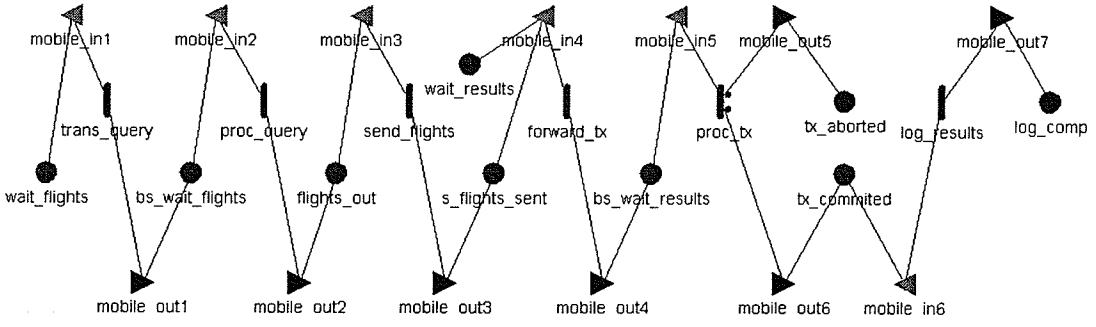


Figure 4.3: SAN submodel of wired communications in the system

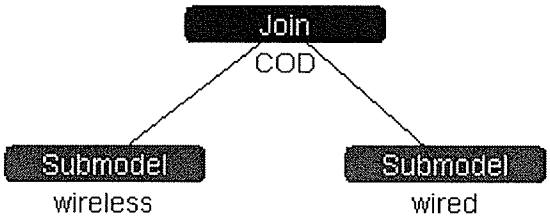


Figure 4.4: SAN composed model of the system

receiving the confirmation message. These performance variables are expressed in *seconds.*

4.1.2 Plugging REV paradigm into the system without handoff

In the Remote Evaluation paradigm, the mobile unit has the code for the execution of the application, but the base station holds the computational power and resources. Once the mobile unit starts the application, it begins to send the code to the base station which then generates the query message and sends it to the server. The UML Sequence Diagram of the system is indicated as Figure 4.6.

Figure 4.7 shows the SAN submodel of the communications between the mobile unit and the base station over the wireless link. In the model the interactions between system components correspond to the timed actions *send_code,return_flights*, *send_tx* and *return_cfmid.* The places *start,code_sent* and *wait_results* represent the states in which the mobile unit starts the application and finishes sending code and transaction messages to the base station. The places *flights_sent* and *end* model the states in which the base station finishes sending the query result and confirmation message of the execution of transactions respectively. The places *loc_mu* and *loc_bs* are used to model different locations of the code during time of migration. The places *s_flights_sent* and *log_comp* represent the shared states in which the server has just sent the query result and transaction log messages to the base station respectively. Figure 4.8 indicates the SAN submodel of the communication between the base station and the server over the wired link. The place *code_sent* represents

Performance Variable	Affecting Models	Reward Function
download_time	wireless	if(wireless->code_sent->Mark()==mu_msgnum+1) return(codereq_size/wireless_speed+code_size/wireless_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum-1) return(qmsg_size/wireless_speed+qmsg_size/wired_speed+ qjob_size/scpu_rate*60+qresults_size/wired_speed+ qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-2)+ txmsg_size/wired_speed*(mu_msgnum-2)+ txjob_size/scpu_rate*60*(mu_msgnum-2)+ logmsg_size/wired_speed*(mu_msgnum-2)+ cfmmsg_size/wireless_speed);

Figure 4.5: Reward functions of performance variables

the shared state in which the mobile unit just finishes sending the code and the base station is going to send the query message to the server. Furthermore, the SAN composed model of the system is shown as Figure 4.9.

The reward function for each performance variable used to measure the performance of the system embedded with the REV paradigm is indicated in Figure 4.10. The performance variable *download_time* represents the time of the base station to load the code from the mobile unit. Because the mobile unit sends the code to the base station directly as soon as it starts the application, the value of *download_time* should be the result of code size divided by wireless link speed.

4.1.3 Plugging MA paradigm into the system without handoff

When the Mobile Agent paradigm is plugged into the system, the mobile unit generates a mobile agent and sends it to the server for execution through the base station. Thus, the server processes the query which is produced by the mobile agent executing on the server,

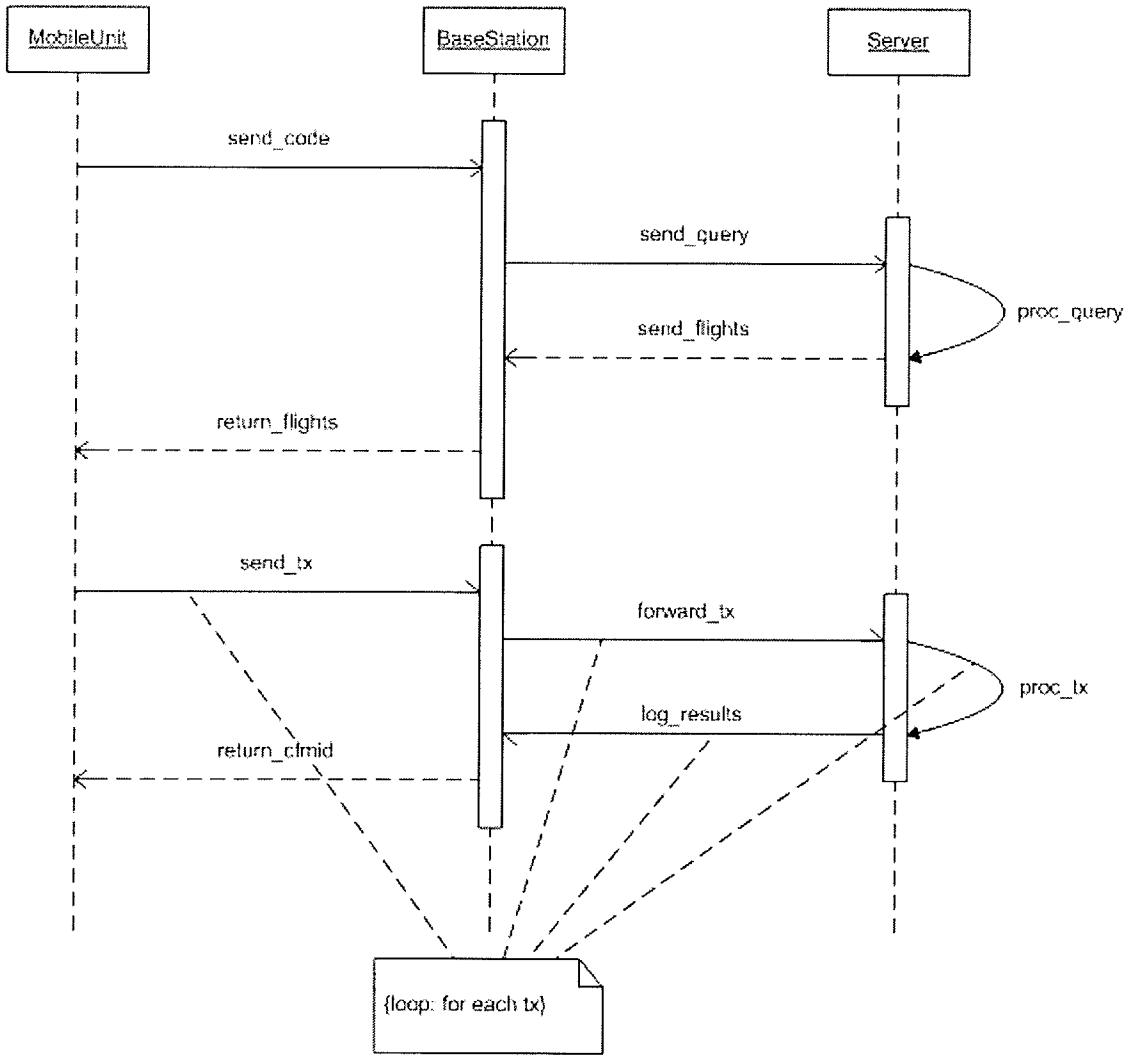


Figure 4.6: Sequence diagram of plugging REV paradigm into the system without handoff

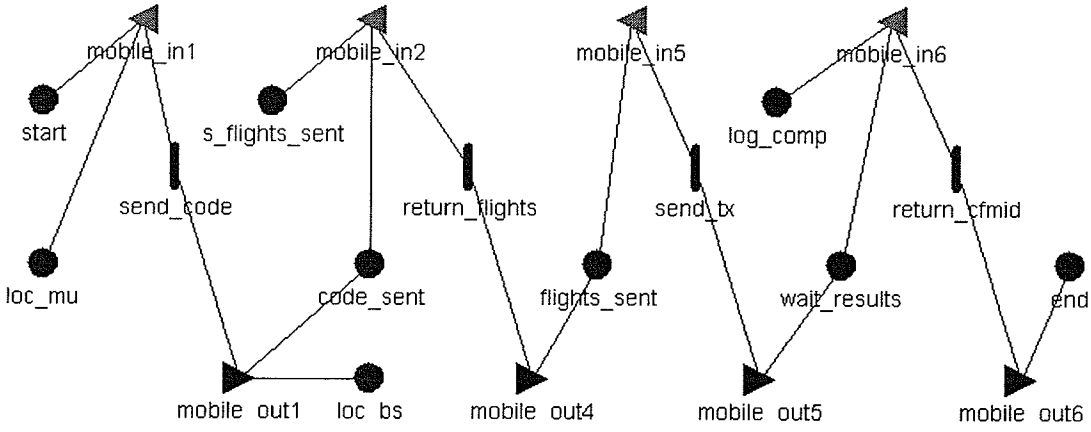


Figure 4.7: SAN submodel of wireless communications in the system

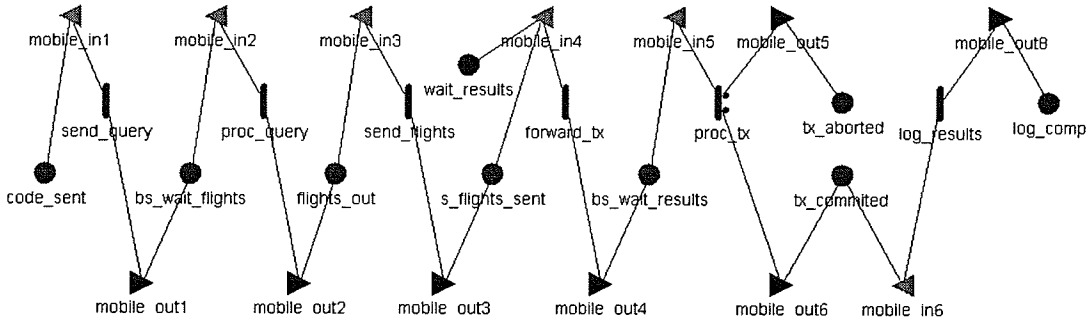


Figure 4.8: SAN submodel of wired communications in the system

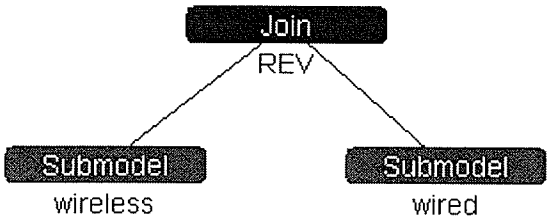


Figure 4.9: SAN composed model of the system

Performance Variable	Affecting Models	Reward Function
download_time	wireless	if(wireless->code_sent->Mark()==mu_msgnum-1) return(code_size/wireless_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum) return(qmsg_size/wired_speed+qjob_size/scpu_rate*60+ qresults_size/wired_speed+ qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-1)+ txmsg_size/wired_speed*(mu_msgnum-1)+ txjob_size/scpu_rate*60*(mu_msgnum-1)+ logmsg_size/wired_speed*(mu_msgnum-1)+ cfmmsg_size/wireless_speed);

Figure 4.10: Reward functions of performance variables

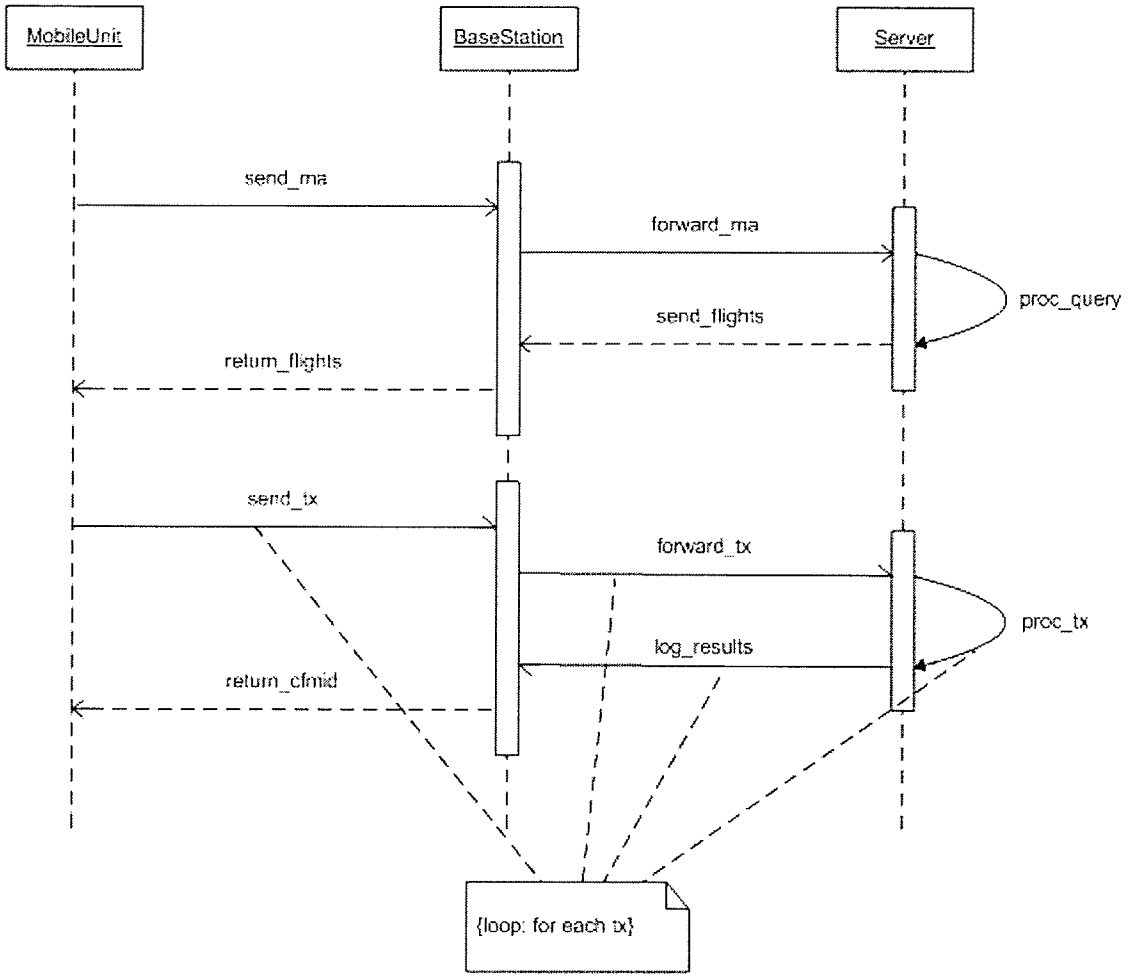


Figure 4.11: Sequence diagram of plugging MA paradigm into the system without handoff

and it will deliver the query result back to the mobile unit. Figure 4.11 illustrates the UML Sequence Diagram of the system.

The SAN submodel of the wireless communications between the mobile unit and the base station for the system embedded with the MA paradigm is represented in Figure 4.12. In the model the interactions between system components correspond to the timed actions *send_ma,return_flights,send_tx* and *return_cfmid*. The SAN submodel of the system expressing the communications over the wired link between the base station and the server is represented in Figure 4.13. The mobile agent travels from the mobile unit to the server via the base station; the places *loc_mu,loc_bs* and *loc_server* model the locations of the mobile

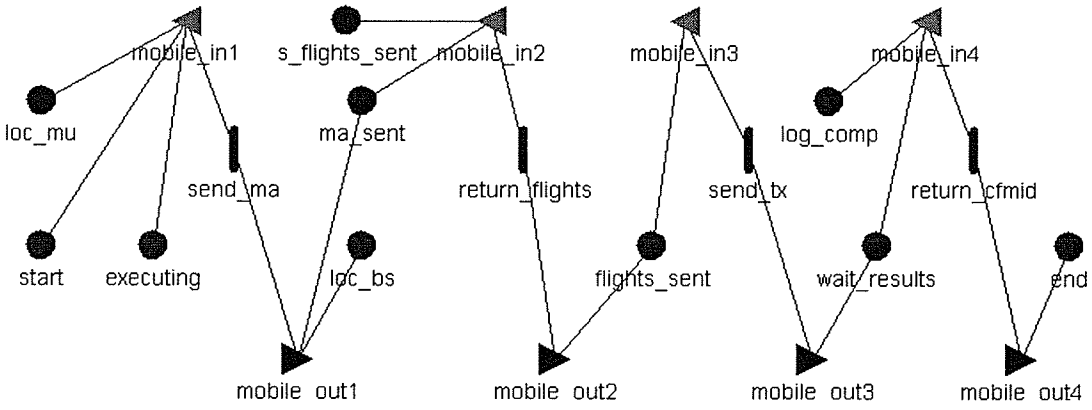


Figure 4.12: SAN submodel of wireless communications in the system

agent. The place *executing* represents the state in which the mobile agent is executing; thus, this state occurs before the mobile agent is sent to the base station and after it arrives at the server. Furthermore, the timed action *proc_query* is enabled by the input gate *mobile_in2* only when the mobile agent is forwarded to the server and its execution is resumed on it. The query result will be delivered back to the mobile unit, represented by the timed actions *send_flights* and *return_flights*. The process of transmission and execution of transactions is the same as it in the system embedded with other code mobility paradigms. Figure 4.14 shows the SAN composed model of the system with the MA paradigm.

Figure 4.15 indicates the reward functions of performance variables for performance evaluation of the system with MA paradigm. Since the mobile agent travels from the mobile unit to the server and executes there, the performance variable *download_time* means the time of the migration of the mobile agent. It should be the sum of the time that the mobile agent travels via the wireless link and the wired link. The performance variable *query_time* should only consider the time for the server to process the query and return the query result back to the mobile unit.

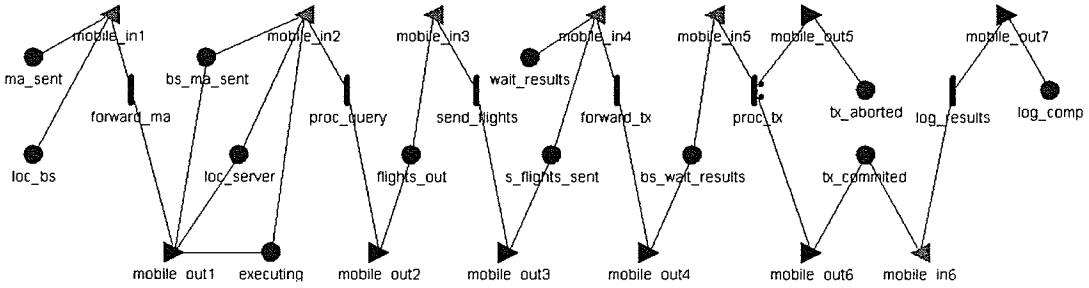


Figure 4.13: SAN submodel of wired communications in the system

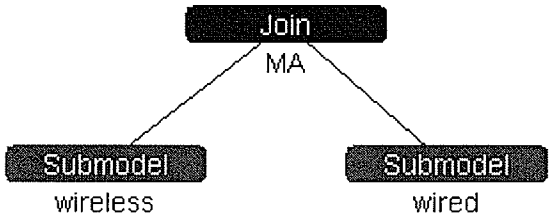


Figure 4.14: SAN composed model of the system

Performance Variable	Affecting Models	Reward Function
download_time	wireless wired	if(wired->bs_ma_sent->Mark()==mu_msgnum+1) return(ma_size/wireless_speed+ma_size/wired_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum) return(qjob_size/scpu_rate*60+ qresults_size/wired_speed+ qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-1)+ txmsg_size/wired_speed*(mu_msgnum-1)+ txjob_size/scpu_rate*60*(mu_msgnum-1)+ logmsg_size/wired_speed*(mu_msgnum-1)+ cfmmsg_size/wireless_speed);

Figure 4.15: Reward functions of performance variables

4.2 Adaptation to the system with handoff

In this case, the mobile unit moves to another cell managed by a new base station while it is working, and it is assumed to move after all the transaction messages have been forwarded to the server. In this section, the adaptation of three code mobility paradigms to the system with handoff will be introduced.

4.2.1 Plugging COD paradigm into the system with handoff

When plugging the Code On Demand paradigm into the system with handoff, the mobile unit sends the code request to the base station which keeps the code and will return it to the mobile unit after receiving the request. After receiving the code, the mobile unit sends the query message to the server, and the server will process the query and return the result. Then the mobile unit generates some transaction messages upon the most desired flight and send them to the server, forwarded by the base station. The mobile unit is assumed to move to a new cell only when the server has received all the transaction messages, and it must send the message of its location information to the old base station. The following events taking place are the same as those in the pre-existing model of the system with handoff. The UML Sequence Diagram of the system is represented in Figure 4.16.

Similarly, the SAN system model is divided into two SAN submodels, indicated in Figure 4.17 and Figure 4.18. Since the COD paradigm is plugged in the pre-existing SAN model of the mobile system with handoff and the code loading only occurs over the wireless link, the SAN submodel represented in Figure 4.18 is the same as the submodel of wired communications of the pre-existing SAN system model shown as Figure 3.7 except the markings of the system. The SAN model of the COD paradigm is plugged into the SAN submodel of wireless communications of the pre-existing SAN model of the system with handoff before the timed action *send_query*, shown in Figure 4.17. Figure 4.19 represents the SAN composed model of the system with handoff embedded with COD paradigm.

Figure 4.20 represents the reward functions of performance variables for simulations of the system with handoff plugged with the COD paradigm. This system and the system

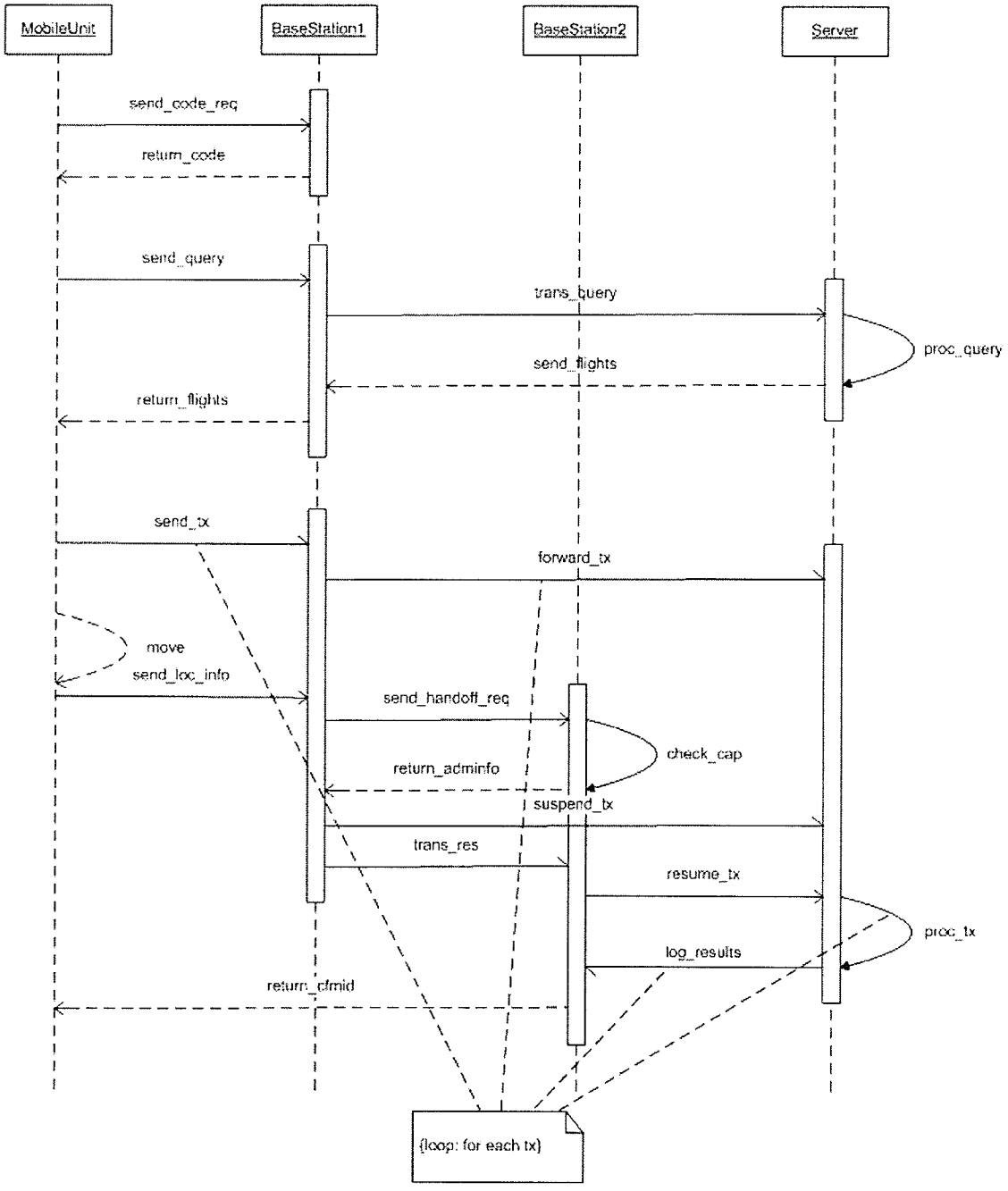


Figure 4.16: Sequence diagram of plugging COD paradigm into the system with handoff

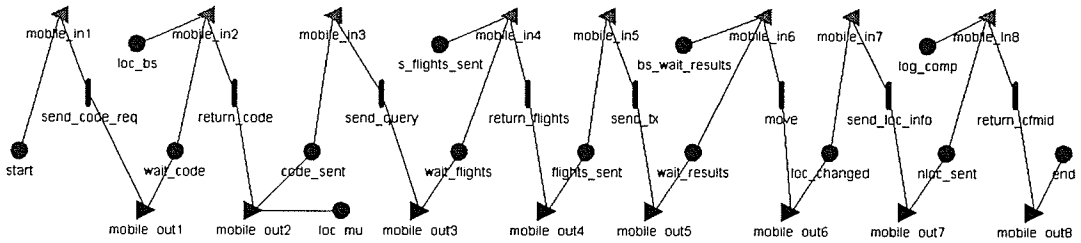


Figure 4.17: SAN submodel of wireless communications in the system

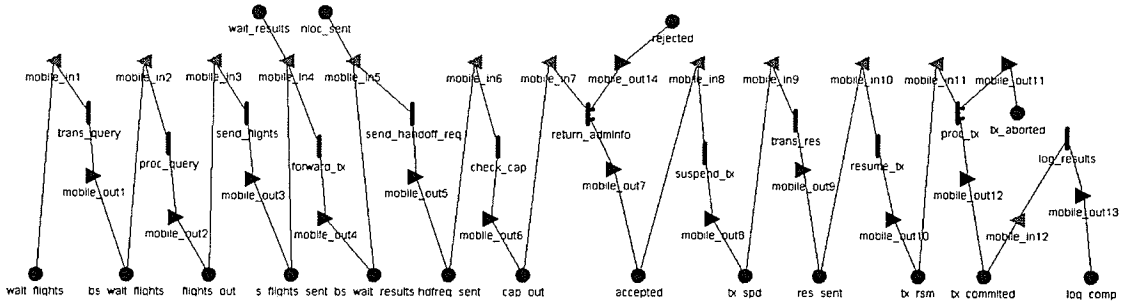


Figure 4.18: SAN submodel of wired communications in the system

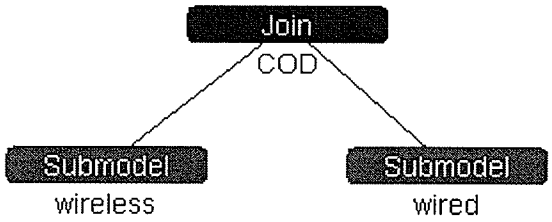


Figure 4.19: SAN composed model of the system

Performance Variable	Affecting Models	Reward Function
download_time	wireless	if(wireless->code_sent->Mark()==mu_msgnum+2) return(codereq_size/wireless_speed+code_size/wireless_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum-1) return(qmsg_size/wireless_speed+qmsg_size/wired_speed+ qjob_size/scpu_rate*60+qresults_size/wired_speed+ qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-3)+ txmsg_size/wired_speed*(mu_msgnum-3)+ locinfo_size/wireless_speed+hdfreq_size/wired_speed+ 0.005+0.001+adminfo_size/wired_speed+ ctrltx_size/wired_speed*2+res_size/wired_speed+ txjob_size/scpu_rate*60*(mu_msgnum-3)+ logmsg_size/wired_speed*(mu_msgnum-3)+ cfmmsg_size/wireless_speed);

Figure 4.20: Reward functions of performance variables

without handoff have the same course of code loading and query processing; thus, only the reward function of the performance variable *tx_transaction_time* differs. It has to consider and calculate the time of the mobile unit's migration, the time of transferring the messages of the mobile unit's location, the old base station's handoff request, the new base station's admission, the transaction control and the handoff resources, and the CPU time of the new base station to check its capacity. The transaction number should be the result of subtracting 3 from the original messages to be sent by the mobile unit, denoted by *mu_msgnum*, because the mobile unit has to send the code request, the query message, the information of its location and the transaction messages.

4.2.2 Plugging the REV paradigm into the system

When the REV paradigm is plugged into the system with handoff, the mobile unit sends the code to the base station for execution, and the base station generates the query message

and sends it to the server. Afterward, the server will process the query and return the result to the mobile unit, forwarded by the base station. The following message exchanges are presented in the sequence diagram of the pre-existing model of the system with handoff. Figure 4.21 shows the UML Sequence Diagram of the system with handoff plugged with the REV paradigm.

The SAN model of the REV paradigm is plugged into the pre-existing SAN model of the system with handoff. In Figure 4.22 which indicates the SAN submodel of wireless interactions in the system, the place *code_sent* model the state in which the mobile unit has just sent the code to the base station, and this state is the shared state also emerging in the SAN submodel of wired communications, shown in Figure 4.23. The timed action *send_query* is enabled by the input gate *mobile_in1* while the system is in the state *code_sent*, which means the base station sends the query message to the server after receiving the code. These two SAN submodels are joined together to express the whole system, represented by the SAN composed model in Figure 4.24.

Figure 4.25 represents the reward functions of performance variables for simulations of the system with handoff plugged with the REV paradigm. The performance variable *download_time* should be the result of the code size divided by the wireless link speed because the code loading only involves transferring code from the mobile unit to the base station. The performance variable *query_time* takes into account the time of sending the query message to the server, the CPU time of the server to process the query, and the time of returning the query result to the mobile unit. The variable *tx_execution_time* has the same reward function as the same variable for simulations of other systems with handoff because these systems are different only before the server begins to process the query.

4.2.3 Plugging MA paradigm into the system

Figure 4.26 represents the UML Sequence Diagram of the system with handoff plugged with the MA paradigm. The mobile unit generates the mobile agent and sends it to the base station which will forward it to the server. The mobile agent executes on the server

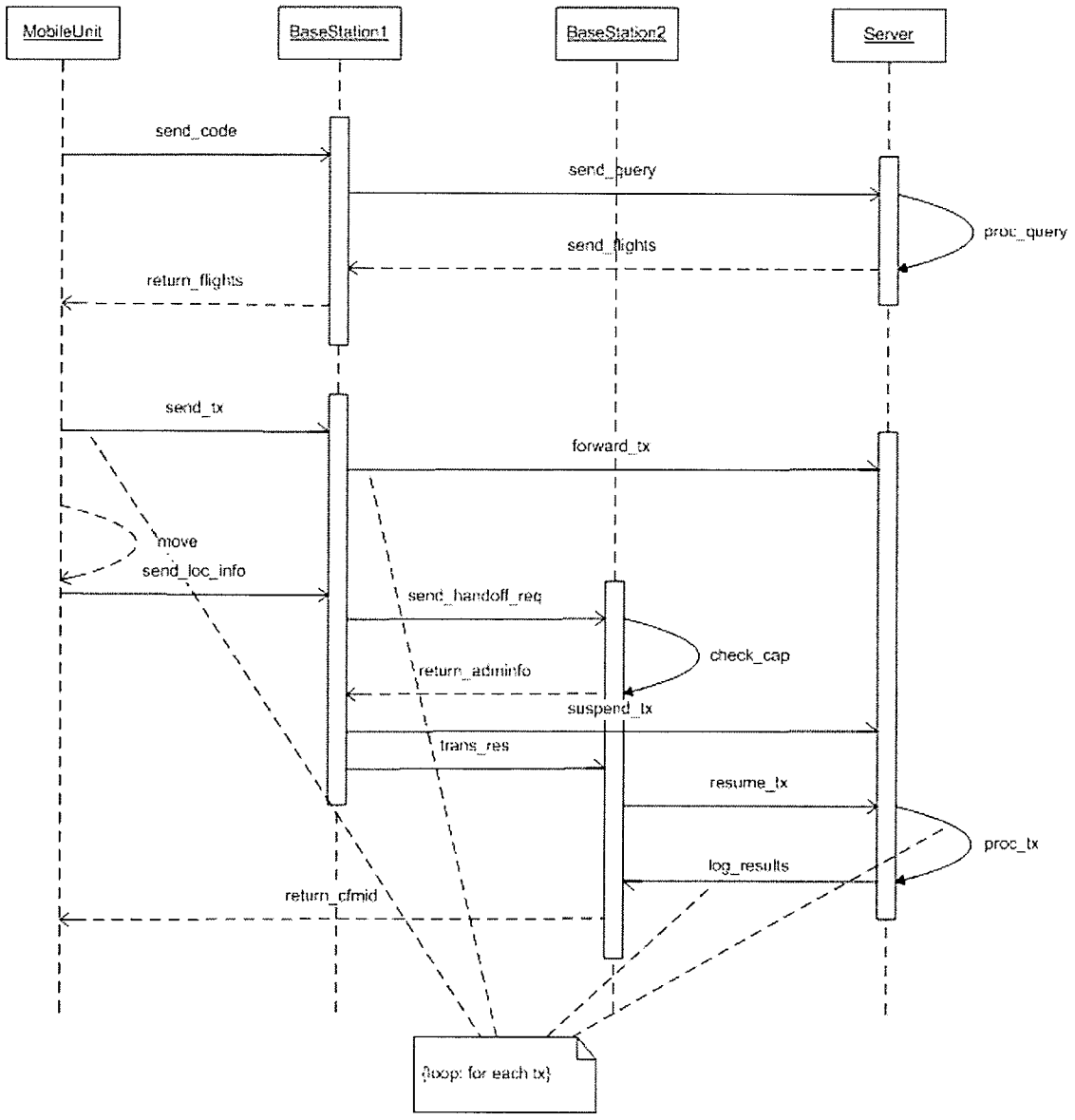


Figure 4.21: Sequence diagram of plugging REV paradigm into the system with handoff

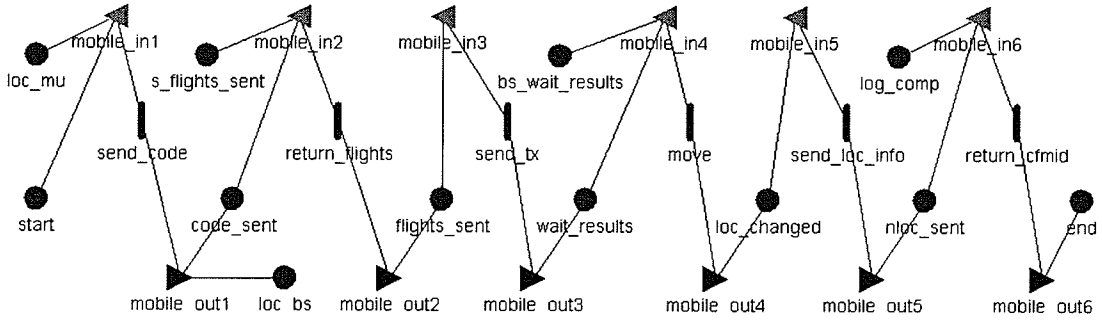


Figure 4.22: SAN submodel of wireless communications in the system

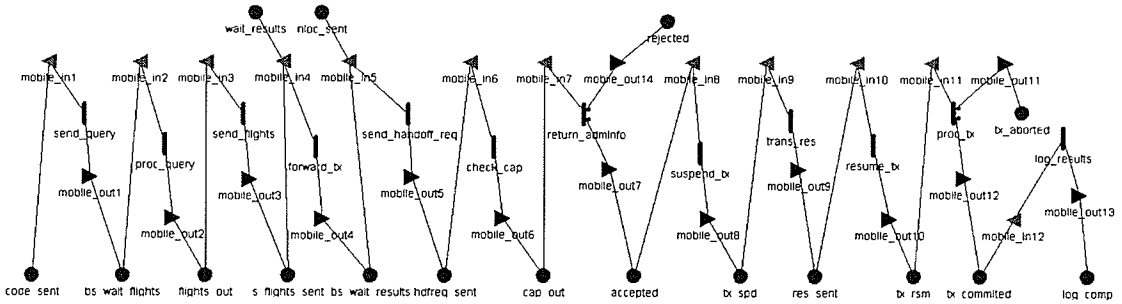


Figure 4.23: SAN submodel of wired communications in the system

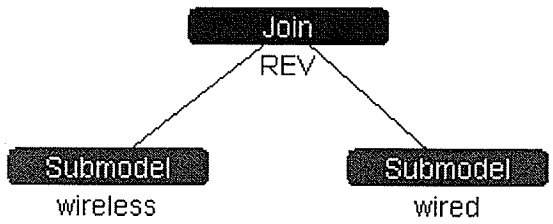


Figure 4.24: SAN composed model of the system

Performance Variable	Affecting Models	Reward Function
download_time	wireless	if(wireless->code_sent->Mark()==mu_msgnum-1) return(code_size/wireless_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum+1) return(qmsg_size/wired_speed+qjob_size/scpu_rate*60+ qresults_size/wired_speed+qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-2)+ txmsg_size/wired_speed*(mu_msgnum-2)+ locinfo_size/wireless_speed+hdfreq_size/wired_speed+ 0.005+0.001+adminfo_size/wired_speed+ctrltx_size/wired_speed*2+ res_size/wired_speed+txjob_size/scpu_rate*60*(mu_msgnum-2)+ logmsg_size/wired_speed*(mu_msgnum-2)+ cfmmsg_size/wireless_speed);

Figure 4.25: Reward functions of performance variables

and produces the query which is processed by the server. The result will be returned to the mobile unit. The following message exchanges are the same as those in the systems with handoff plugged with other code mobility paradigms.

Figure 4.27 and Figure 4.28 are the SAN submodels of the system; the former models the messages between the system components carried by the wireless link, and the latter models the messages carried by the wired link. The SAN model of the MA paradigm is divided into two parts and plugged into two pre-existing SAN submodels of the system with handoff apart, which is because the mobile agent is transferred over the wireless and wired link. The places *loc_bs* and *ma_sent* model two shared states in which the mobile agent has sent the mobile agent to the base station and the code is located on the base station. The input gate *mobile_in2* enables the timed action *proc_query* only when the base station has forwarded the mobile agent to the server and the execution of the mobile agent is resumed, represented by the places *bs_ma_sent*, *loc_server* and *executing*. Figure 4.29 represents the SAN composed model of the system with handoff plugged with the MA paradigm.

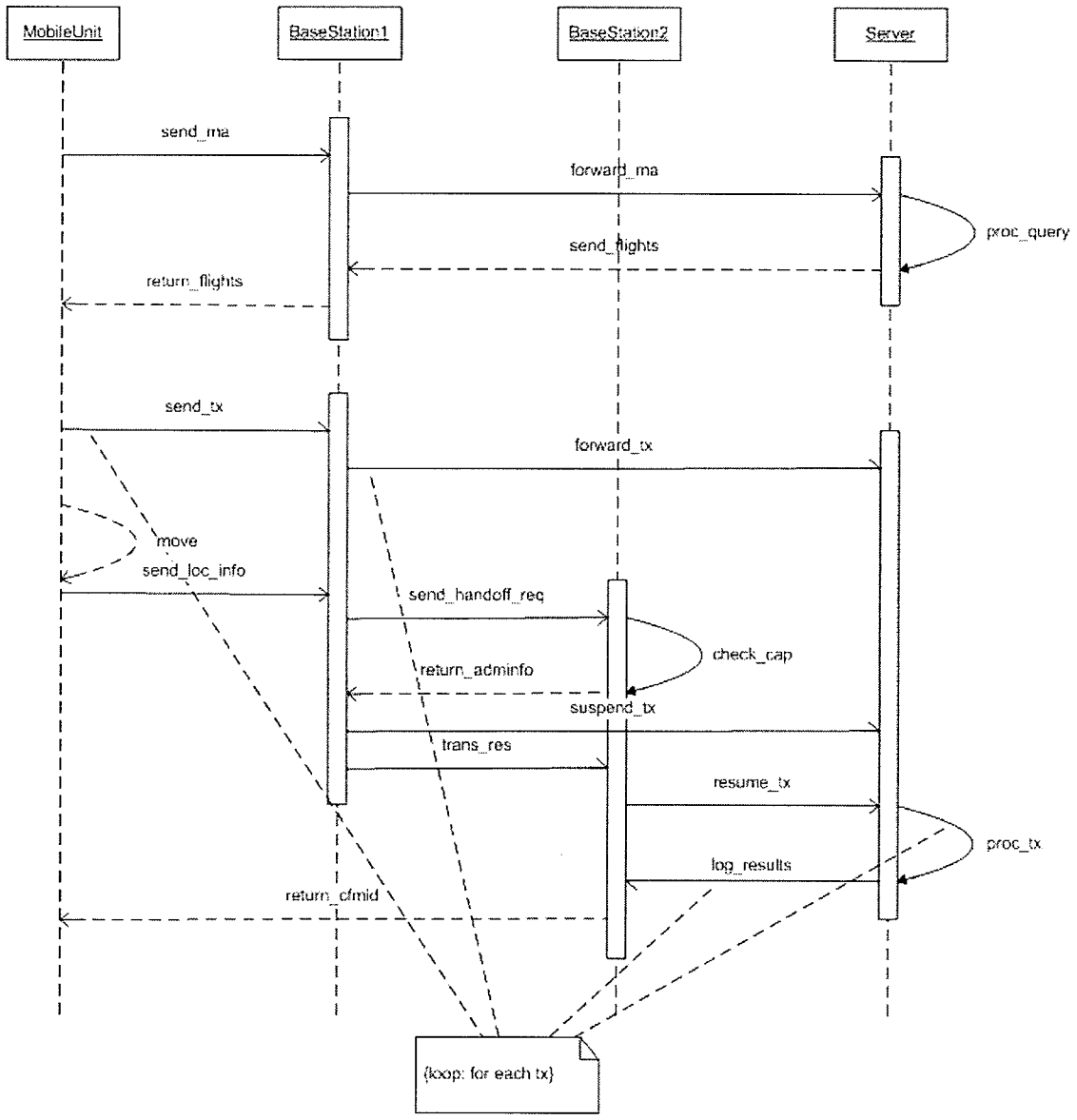


Figure 4.26: Sequence diagram of plugging MA paradigm into the system with handoff

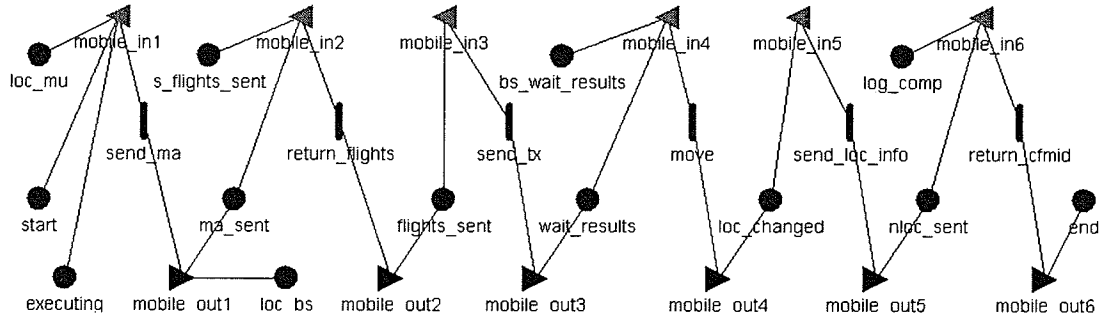


Figure 4.27: SAN submodel of wireless communications in the system

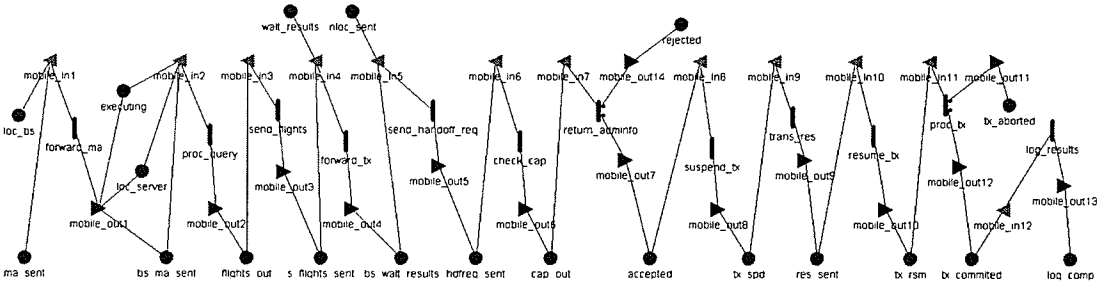


Figure 4.28: SAN submodel of wired communications in the system

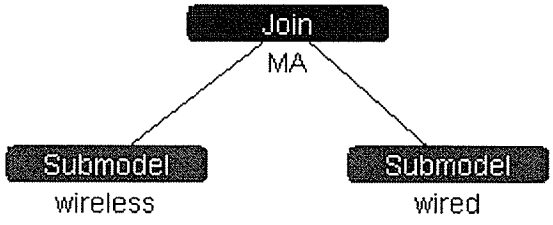


Figure 4.29: SAN composed model of the system

Performance Variable	Affecting Models	Reward Function
download_time	wireless wired	if(wired->bs_ma_sent->Mark()==mu_msgnum+2) return(ma_size/wireless_speed+ma_size/wired_speed);
query_time	wireless wired	if(wireless->flights_sent->Mark()==mu_msgnum+1) return(qjob_size/scpu_rate*60+qresults_size/wired_speed+ qresults_size/wireless_speed);
tx_execution_time	wireless wired	if(wireless->end->Mark()==0) return(txmsg_size/wireless_speed*(mu_msgnum-2)+ txmsg_size/wired_speed*(mu_msgnum-2)+ locinfo_size/wireless_speed+hdfreq_size/wired_speed+ 0.005+0.001+adminfo_size/wired_speed+ctrltx_size/wired_speed*2+ res_size/wired_speed+txjob_size/scpu_rate*60*(mu_msgnum-2)+ logmsg_size/wired_speed*(mu_msgnum-2)+ cfmmsg_size/wireless_speed);

Figure 4.30: Reward functions of performance variables

Figure 4.30 illustrates the reward functions of performance variables needed for simulations and the performance evaluation of the system. The reward function of the variable *download_time* calculates the time of transferring the mobile agent from the mobile unit to the server, the reward function of the variable *query_time* calculates the time of processing the query and returning the result to the mobile unit, and the reward function of the variable *tx_execution_time* calculates the time of the mobile unit from sending the transaction messages to receiving the confirmation message.

Chapter 5

Numerical Results

As we have all of the prepared SAN models of the systems, they can be solved by the Mobius solver to get the results of performance variables for the performance evaluation of the system. Because the systems are grouped into two categories which are the systems without handoff and those with handoff, the results of simulations will be assessed for two cases respectively. In order to evaluate the system performance with several different parameter values, a range is defined for one parameter in each simulation. Three simulations are executed for each model by varying three different parameter values respectively. For each simulation the results of the performance variables *download_time, query_time* and *transaction_execution_time* are indicated in one chart. All the results produced by the simulations are gathered to compare the performance of systems with different code mobility paradigms.

5.1 Result assessments for the systems without handoff

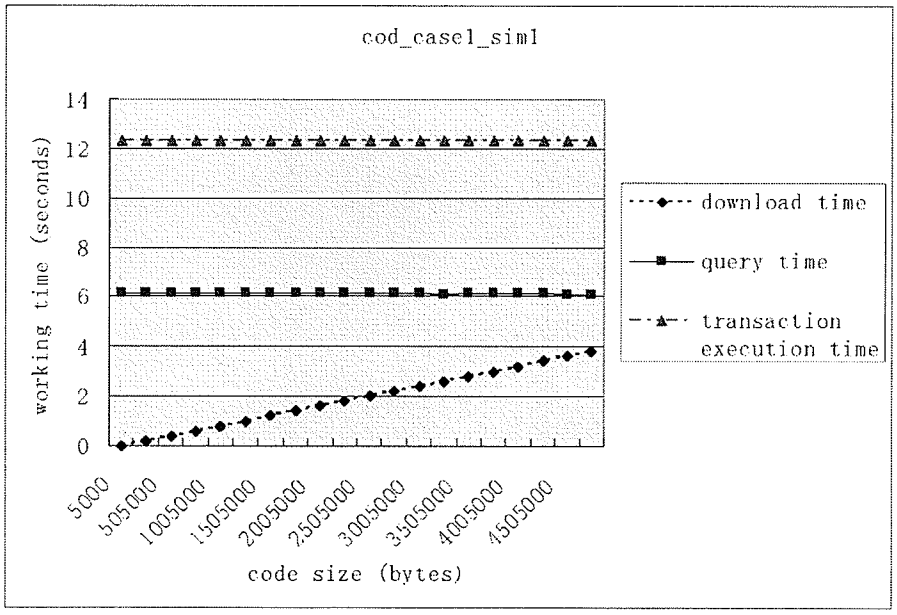
Figure 5.1(a) indicates a set of parameter values for the simulation of the system without handoff plugged with the COD paradigm when varying the size of the code transferred between the mobile unit and the base station, denoted by *code_size*. The performance

variable *download_time* evaluates the time of the mobile unit to load the code from the base station. The sizes of all the messages exchanged between different system components are expressed in *bytes*, the CPU rate is expressed in *jobs/min*, and the wireless and wired link speed are expressed in *bytes/sec*. For the system plugged with the COD paradigm the number of transaction messages should be the result of subtracting 2 from the number of the original messages to be sent by the mobile unit, hence, the transaction message number for the simulation of the system should be 8. The results from solution of the model of the system without handoff plugged with the COD paradigm by varying the code size are shown in Figure 5.1(b). The results of the performance variable *download_time* increases linearly when the value of the parameter *code_size* increases. The figure shows that the results of the performance variables *query_time* and *tx_execution_time* do not have visible changes with the variation of the code size because the code size has no impact on them.

Figure 5.2(a) illustrates a group of parameter values used for the simulation of the system without handoff plugged with the COD paradigm when varying the number of transaction messages sent from the mobile unit to the server. The performance variable *tx_execution_time* represents the time of the completion of all the transactions initiated by the mobile unit, and it includes the time of sending the transaction messages from the mobile unit to the server, processing the transactions on the server and returning the confirmation message to the mobile unit. The parameter value *mu_msgnum* increases from 3 to 98,which means the number of transaction messages increases from 1 to 96. The results from solution of the model of the system without handoff plugged with the COD paradigm by varying the transaction message number are indicated in Figure 5.2(b), and it shows that the time of completing the execution of all the transactions increases with the increments of the number of transaction messages. Furthermore, the size of the transaction jobs to be processed by the server should be consonant with the number of transaction messages normally, however, the transaction job size is assumed to be fixed in this work no matter how the transaction message number changes so the results only show the impact of the number of transaction messages to the performance variables.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.

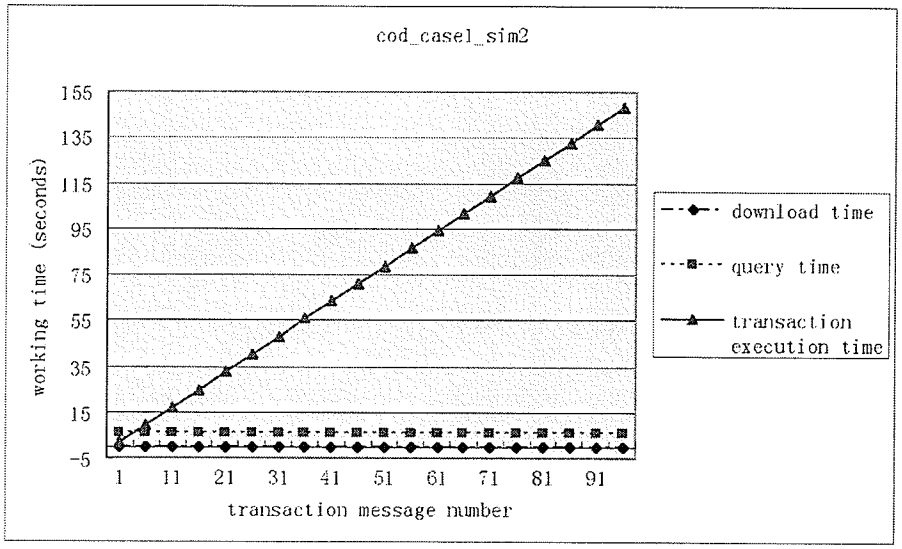


(b) Results from solution of the model.

Figure 5.1: Simulation of the COD paradigm model without handoff when varying the code size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[3-98] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.2: Simulation of the COD paradigm model without handoff when varying the transaction message number.

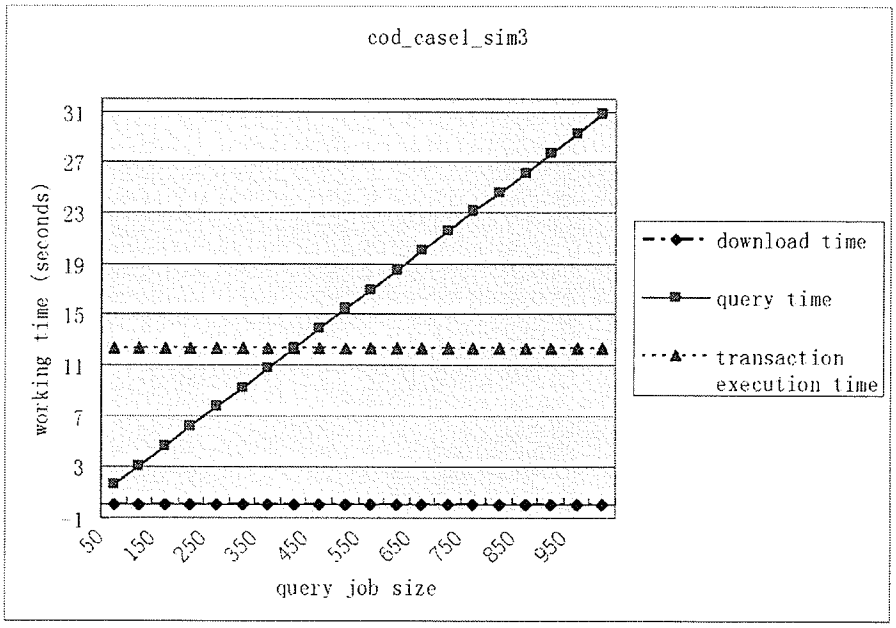
Figure 5.3(a) indicates a group of parameter values used for the simulation of the system without handoff plugged with the COD paradigm when varying the size of the query jobs to be processed by the server, denoted by *qjob_size*. A incremental range is defined for the parameter *qjob_size* so that the results of different performance variables can be calculated. Moreover, the parameter *mu_msgnum* is set as 10 so that the number of transaction messages should be 9 because in this system the mobile unit sends the code and the transaction messages out to the base station. The performance variable *query_time* represents the time of completing the query initiated by the mobile unit, and it includes the time of sending the query message from the mobile unit to the server, processing the query on the server and returning the query results to the mobile unit. The results from solution of the model of the system without handoff plugged with the COD paradigm by varying the query job size are indicated in Figure 5.3(b). The results show that the query time increases with the increments of the query job size and the download time and transaction execution time remain the same.

Figure 5.4(a) shows the parameter values taken into account the simulation of the system without handoff plugged with the REV paradigm when varying the code size. The performance variable *download_time* represents the time of sending the code from the mobile unit to the base station, and an incremental range is set for the parameter *code_size* for the simulation. The number of transaction messages for solving the model should be 8 because it is the result of subtracting 1 from the number of the original messages to be sent by the mobile unit, denoted by *mu_msgnum*. The results from solution of the model of the system without handoff plugged with the REV paradigm by varying the code size are represented in Figure 5.4(b), and it shows that the download time increases linearly with the increments of the code size. However, the size of the code which is loaded by the base station has no effect to the query time and transaction execution time.

Figure 5.5(a) reveals a group of parameter values used for the simulation of the system without handoff plugged with the REV paradigm when varying the number of the transaction messages sent from the mobile unit to the server. In the simulation the parameter

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.

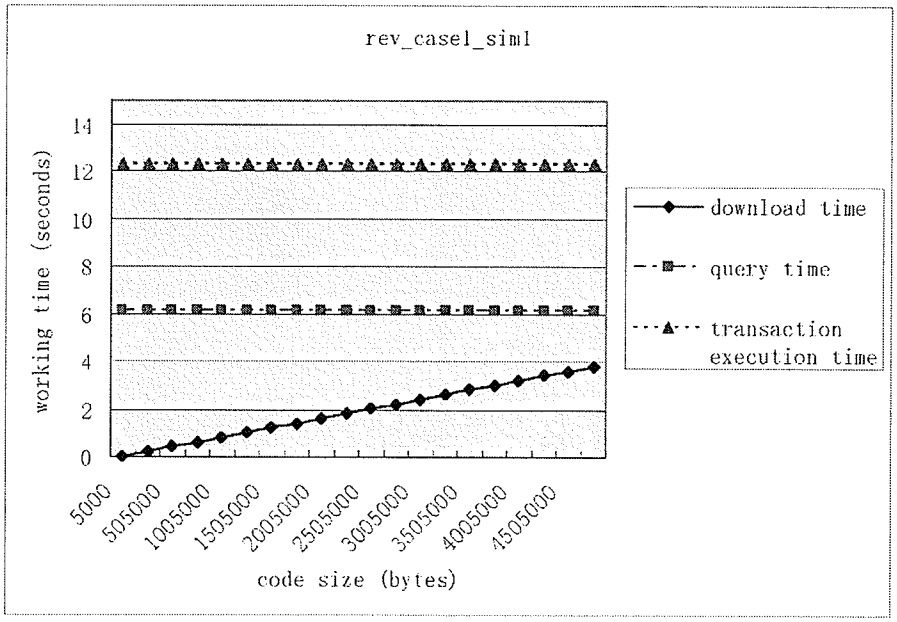


(b) Results from solution of the model.

Figure 5.3: Simulation of the COD paradigm model without handoff when varying the query job size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	9
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.4: Simulation of the REV paradigm model without handoff when varying the code size.

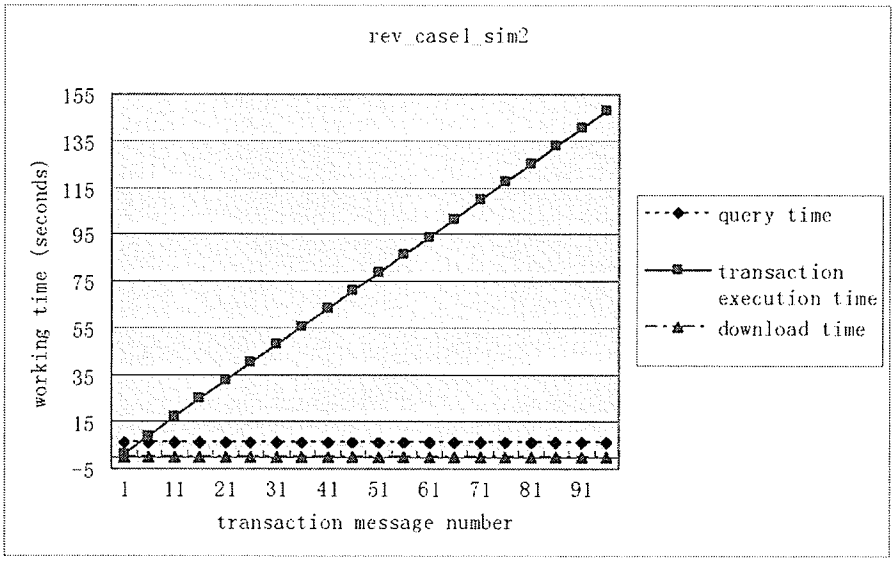
mu_msgnum increases from 2 to 97, which means the number of transaction messages increases from 1 to 96, and the code size is set as 50000 bytes. The results from solution of the model of the system without handoff plugged with the REV paradigm by varying the transaction message number are indicated in Figure 5.5(b) which shows the effect the transaction messages number has to the transaction execution time. The chart also shows in the system the download time is less than 1 second and the query time is about 6 seconds.

Figure 5.6(a) displays a set of parameter values employed for the simulation of the system without handoff plugged with the REV paradigm when varying the size of query jobs processed by the server. A incremental range is defined for the parameter *qjob_size* in the simulation of the system. The query time evaluates the time of completing the query started on the base station, including the time of sending the query message from the base station to the server, processing the query on the server and returning the query results from the server to the mobile unit. The results from solution of the model of the system without handoff plugged with the REV paradigm by varying the query job size are indicated in Figure 5.6(b) which shows the query time increases linearly with the increments of the query job size, and the transaction execution time is about 12 seconds.

Figure 5.7(a) illustrates a set of parameter values taken into account the simulation of the system without handoff plugged with the MA paradigm when varying the size of the mobile agent. The same incremental range is defined for the parameter *ma_size* as *code_size*, however, in the real case the size of the mobile agent may be different with the code size. To compare the performance of different code mobility paradigms, other parameter values in simulations are uniform. The results from solution of the model of the system without handoff plugged with the MA paradigm by varying the mobile agent size are represented in Figure 5.7(b). The download time denotes the time of sending the mobile agent from the mobile unit to the server for execution, and it increases linearly from less than 1 second to 4 seconds with the increments of the mobile agent size between the range of 5000 to 4755000 bytes. Nevertheless, the mobile agent size shows no impact on the query time and the transaction execution time.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[2-97] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.

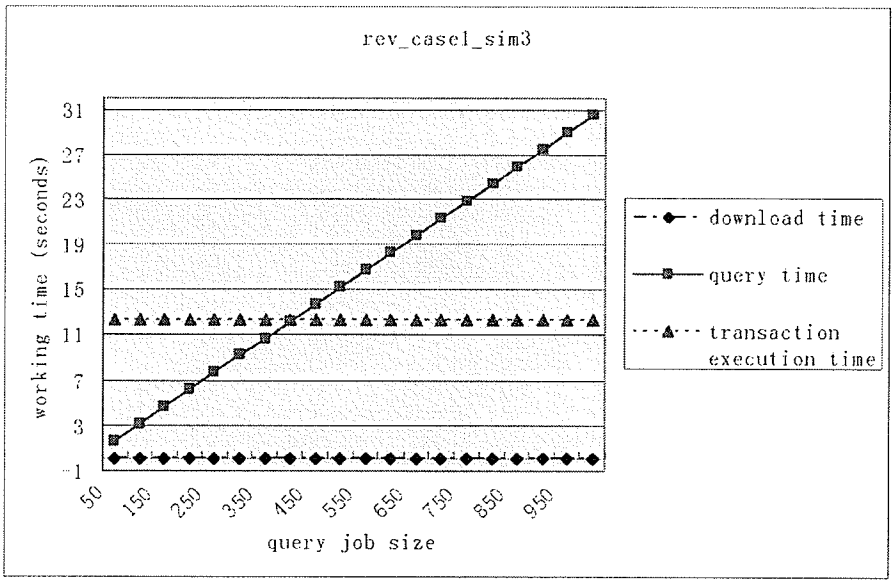


(b) Results from solution of the model.

Figure 5.5: Simulation of the REV paradigm model without handoff when varying the transaction message number.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	9
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.

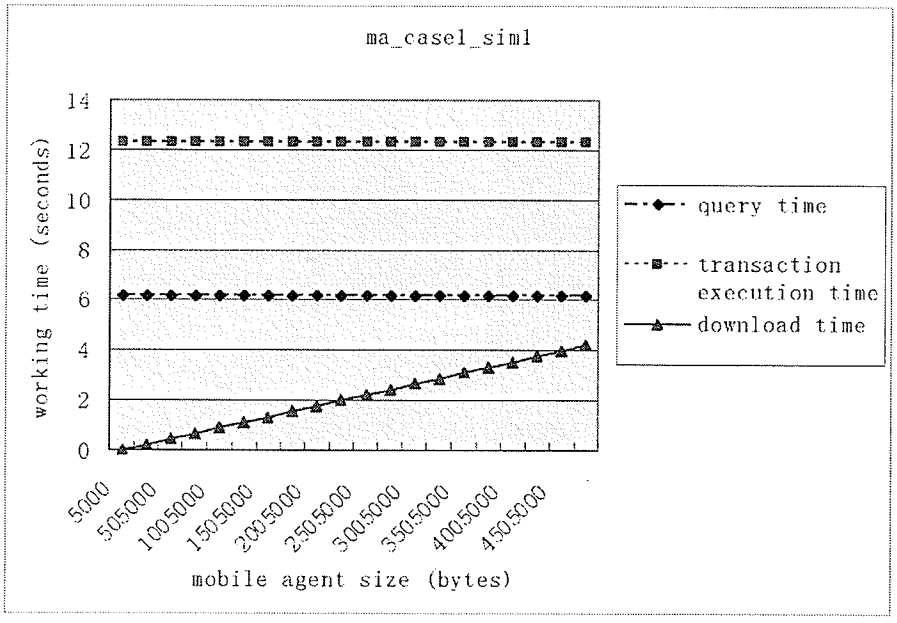


(b) Results from solution of the model.

Figure 5.6: Simulation of the REV paradigm model without handoff when varying the query job size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	9
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.7: Simulation of the MA paradigm model without handoff when varying the mobile agent size.

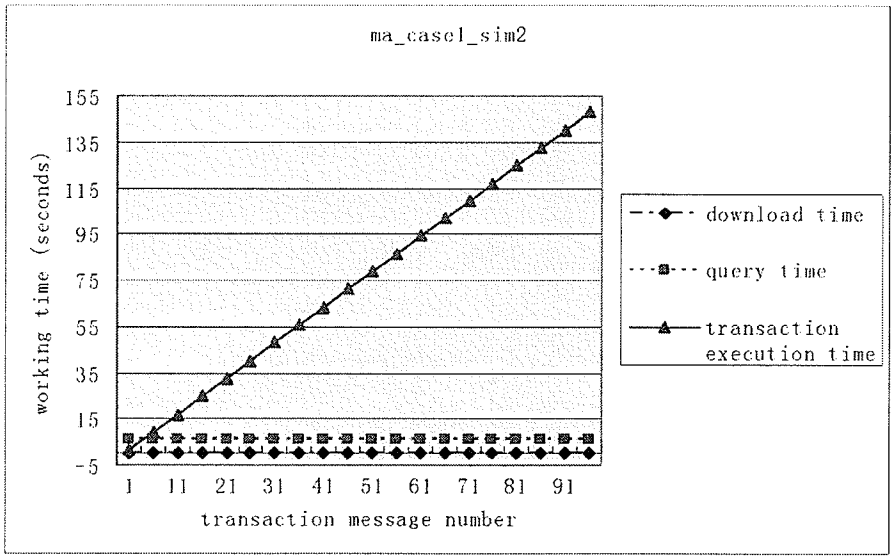
Figure 5.8(a) reveals a group of parameter values defined for the simulation of the system without handoff plugged with the MA paradigm when varying the number of transaction messages sent from the mobile unit to the server. In the simulation the size of the mobile agent is defined as 50000 bytes, the same as the code size in simulations of the other code mobility paradigms, and the same range is defined for the number of transaction messages. The results from solution of the model of the system without handoff plugged with the MA paradigm by varying the transaction message number are shown in Figure 5.8(b). The transaction execution time denotes the time of completing the execution of transactions, including the time of transferring transaction messages from the mobile unit to the server, processing transactions on the server and returning the confirmation message to the mobile unit, and it relates to the transaction message number. The results show the transaction execution time increases linearly with the increments of the transaction message number which has no effect on the download time and the query time.

Figure 5.9(a) displays a set of parameter values used for the simulation of the system without handoff plugged with the MA paradigm when varying the size of the query job processed by the server. The same incremental range is defined for the parameter *qjob_size*, and the transaction message number is defined as 8. In the system the query time considers the time of processing the query on the server and returning the query results to the mobile unit. The results from solution of the model of the system without handoff plugged with the MA paradigm by varying the query job size are indicated in Figure 5.9(b) which shows the query time increases from 1 to 31 seconds when the query job size rises from 50 to 1000 jobs. However, the transaction execution time is about 12 seconds all through regardless of the query job size.

Figure 5.10 represents the comparison of the total working time of the systems without handoff plugged with different code mobility paradigms when varying the size of the code or the mobile agent, and it reveals that the REV paradigm has the similar performance with the COD paradigm and the MA paradigm obviously has the worst performance while the size of code is assumed to be equal to the size of mobile agent in simulations, which

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[2-97] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.

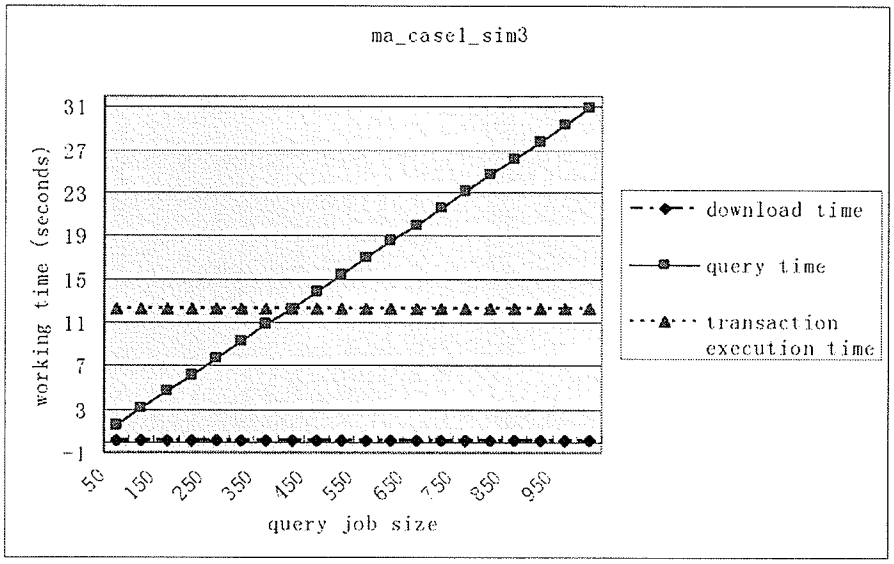


(b) Results from solution of the model.

Figure 5.8: Simulation of the MA paradigm model without handoff when varying the transaction message number.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	9
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.9: Simulation of the MA paradigm model without handoff when varying the query job size.

results from the heavy network workload caused by the large mobile agent size. Moreover, the REV paradigm performs better than the COD paradigm slightly. Figure 5.11 shows the comparison of the total working time of the systems without handoff plugged with three different code mobility paradigms when varying the number of transaction messages. It displays three code mobility paradigms have very similar performance and there are about 1 to 100 milliseconds differences in the total working time of three systems with the changes of the transaction message number in respect that the transaction message number only affects the performance of the transaction execution. The REV paradigm has the slight superiority than the other ones. Figure 5.12 indicates the comparison of the total working time of the systems without handoff plugged with different code mobility paradigms when varying the query job size. It shows three systems have similar performance with the increments of the query job size; however, the system plugged with the REV paradigm still has the shortest working time even though it is not remarkable.

5.2 Result assessments for the systems with handoff

Figure 5.13(a) displays a group of parameter values defined for the simulation of the system with handoff plugged with the COD paradigm when varying the size of the code transferred from the base station to the mobile unit upon the request. All the messages exchanged between system components are expressed in *bytes*, and the wired and wireless link speed are expressed in *bytes/sec*. A incremental range is defined for the code size and the number of transaction messages is defined as 8 for the simulation. The results from solution of the model of the system with handoff plugged with the COD paradigm by varying the code size are indicated in Figure 5.13(b) which shows the download time increases linearly with the increments of the code size, the query time remains about 6 seconds and the transaction execution time is about 12 seconds. The mobile system with handoff and the system without handoff should have the same download and query time, but the former has the larger transaction execution time because it has to handle the handoff and this time difference is only about 20 milliseconds.

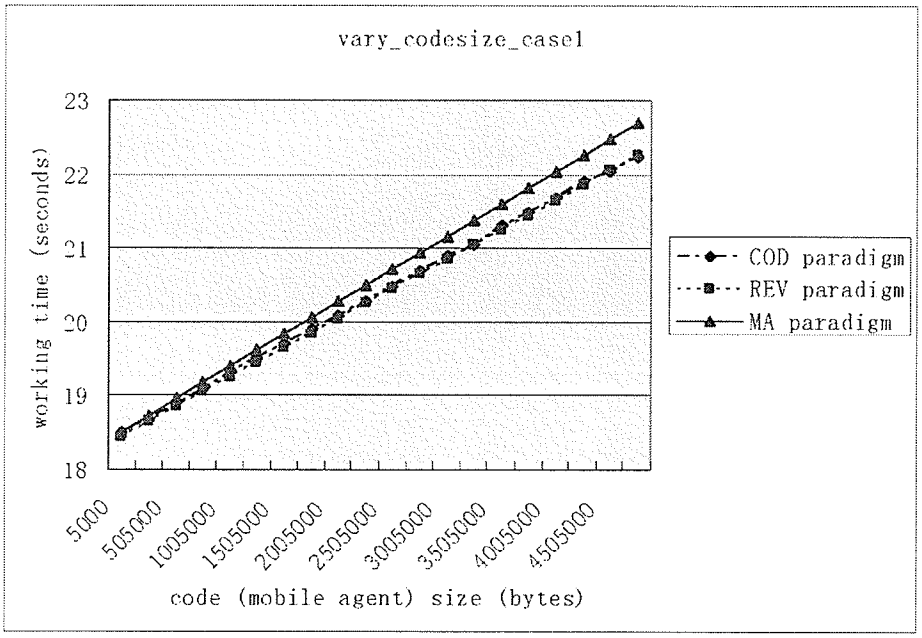


Figure 5.10: Comparison of results when varying the code size.

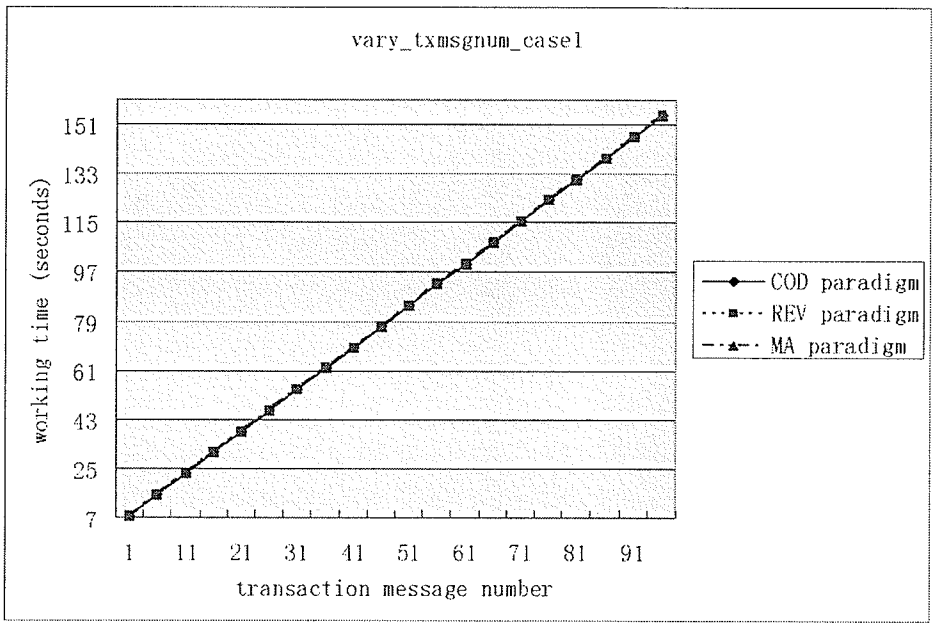


Figure 5.11: Comparison of results when varying the transaction message number.

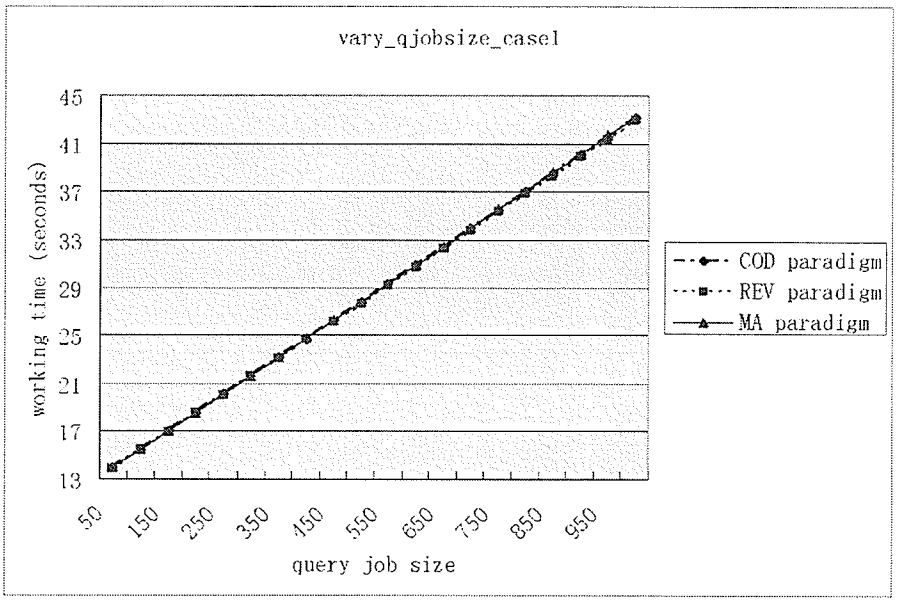
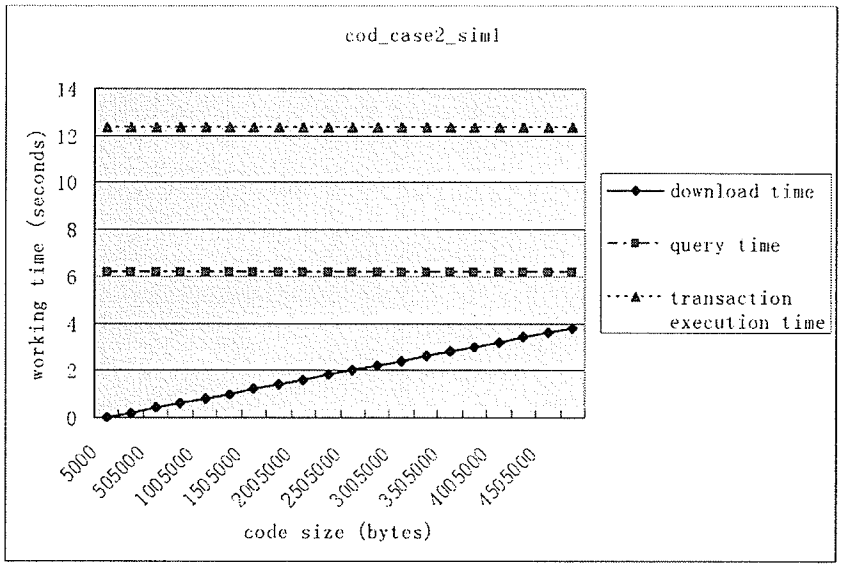


Figure 5.12: Comparison of results when varying the query job size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	11
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.13: Simulation of the COD paradigm model with handoff when varying the code size.

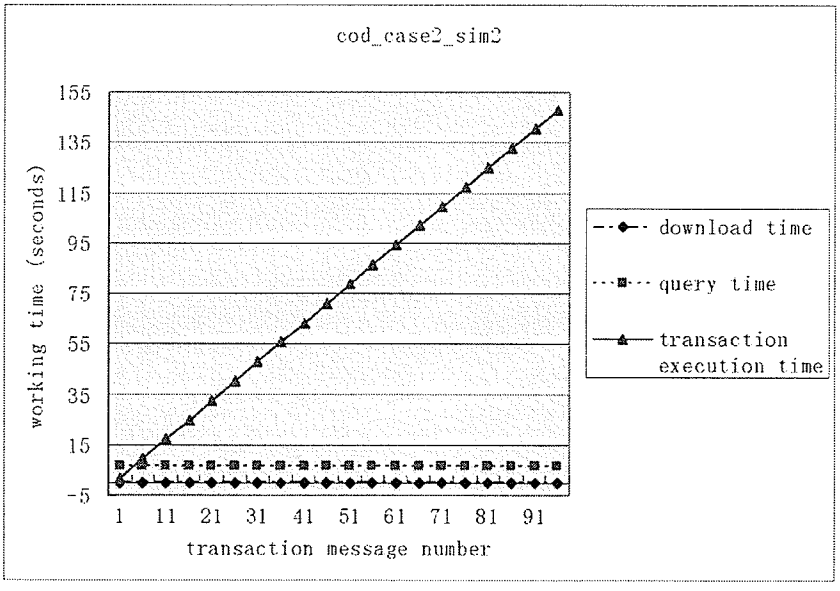
Figure 5.14(a) shows a set of parameter values defined for the simulation of the system with handoff plugged with the COD paradigm when varying the number of transaction messages sent from the mobile unit to the server. The same incremental range is defined for the transaction message number which changes from 1 to 96 in the simulation. The size of the message carrying the information of the mobile unit's location is defined as 1000 bytes, and the messages exchanged between system components over the wired link for handling the handoff are defined as 500 bytes. Furthermore, the size of the resource transferred between two base stations is defined as 200000 bytes. The results from solution of the model of the system with handoff plugged with the COD paradigm by varying the transaction message number are illustrated in Figure 5.14(b) which reveals the transaction execution time increases with the increments of the transaction message number which has no impact on the download time and the query time.

Figure 5.15(a) indicate a set of parameter values defined for the simulation of the system with handoff plugged with the COD paradigm when varying the size of the query jobs processed by the server. The incremental range is defined for the query job size which changes from 50 to 1000, and the number of the messages to be sent by the mobile unit is defined as 11 so that the transaction message number should be 8 because the mobile unit sends the request for code, the query message, the transaction messages and the location notification message. The results from solution of the model of the system with handoff plugged with the COD paradigm by varying the query job size are indicated in Figure 5.15(b) which shows the query job size influences the query time and makes it to vary from 1 second to 31 seconds. The transaction execution time is about 12 seconds and the download time is less than 1 second when the code size is 50000 bytes.

Figure 5.16(a) indicates a batch of parameter values taken into account the simulation of the system with handoff plugged with the REV paradigm when varying the size of the code sent from the mobile unit to the base station for remote execution, and the simulation is taken based on the variation of the code size. The number of transaction messages should be the result of subtracting 2 from the number of the original messages to be sent

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[4-99] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.

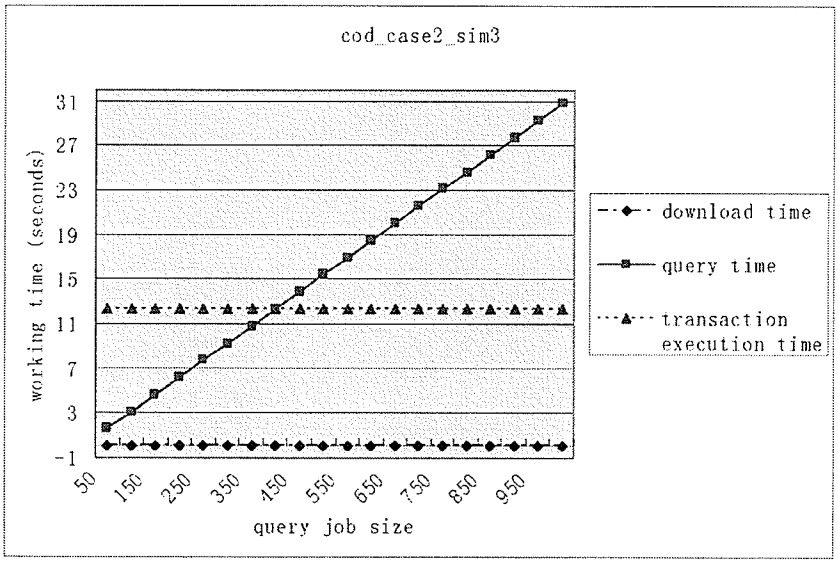


(b) Results from solution of the model.

Figure 5.14: Simulation of the COD paradigm model with handoff when varying the transaction message number.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code request message	codereq_size	500.0
Size of the code to be returned	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	11
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.15: Simulation of the COD paradigm model with handoff when varying the query job size.

by the mobile unit in respect that the mobile unit will send the code, the information of its location and the transaction messages to the base station. The results from solution of the model of the system with handoff plugged with the REV paradigm by varying the code size are illustrated in Figure 5.16(b) which reveals that the system has the same download time and the query time as the system without handoff plugged with the REV paradigm, but the former has the larger transaction execution time since it has to handle the handoff.

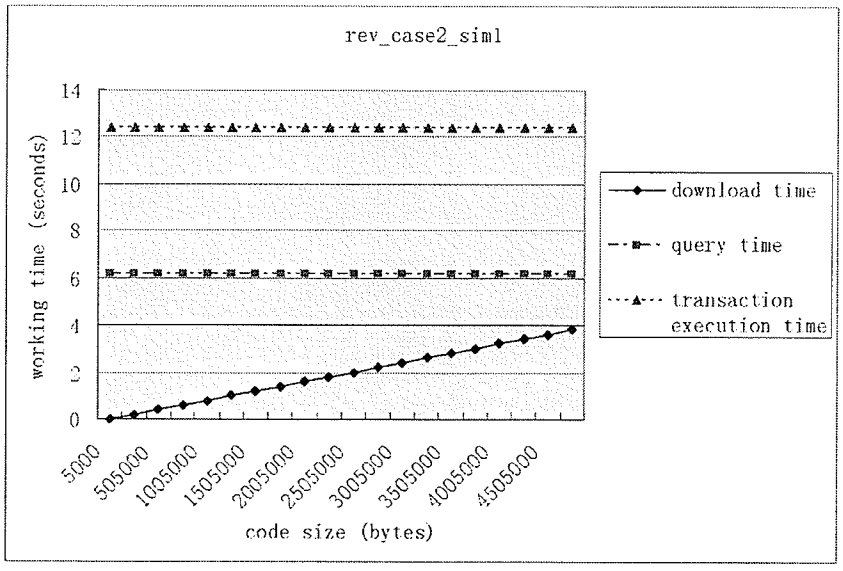
Figure 5.17(a) represents a group of parameter values defined for the simulation of the system with handoff plugged with the REV paradigm when varying the number of transaction messages sent by the mobile unit to the server, and the simulation is taken based on the variation of the transaction message number whose range is defined from 1 to 96 since the mobile unit has to send the code, the transaction messages and the location notification message. The results from solution of the model of the system with handoff plugged with the REV paradigm by varying the transaction message number are shown in Figure 5.17(b) which reveals the transaction execution time increases from 1 second to 148 seconds with the increments of the transaction messages number. The download time is less than 1 second when the code size is defined as 50000 bytes and the query time is about 6 seconds when the query job size is defined as 200.

Figure 5.18(a) indicates a batch of parameter values defined for the simulation of the system with handoff plugged with the REV paradigm when varying the size of query jobs processed by the server, and the simulation is taken based on the variation of the query job size whose range is defined from 50 to 1000 and the transaction message number is defined as 8 since it should be the result of subtracting 2 from the number of messages sent by the mobile unit which is denoted by *mu_msgnum*. The results from solution of the model of the system with handoff plugged with the REV paradigm by varying the query job size are displayed in Figure 5.18(b) which reveals the query time increases from 1 second to 31 seconds with the increments of the query job size and the transaction execution time is around 12 seconds when the transaction message number is defined as 8.

Figure 5.19(a) represents a set of parameter values needed for calculating the results

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.

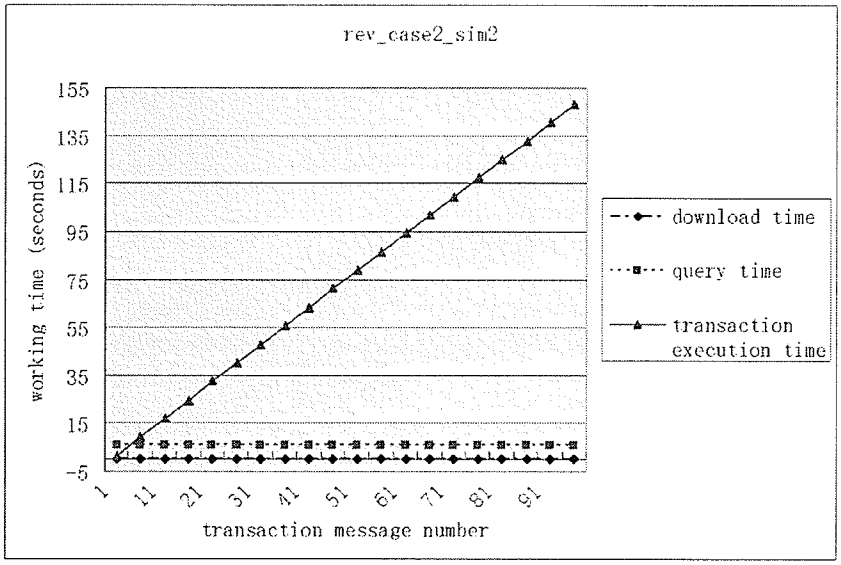


(b) Results from solution of the model.

Figure 5.16: Simulation of the REV paradigm model with handoff when varying the code size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[3-98] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.

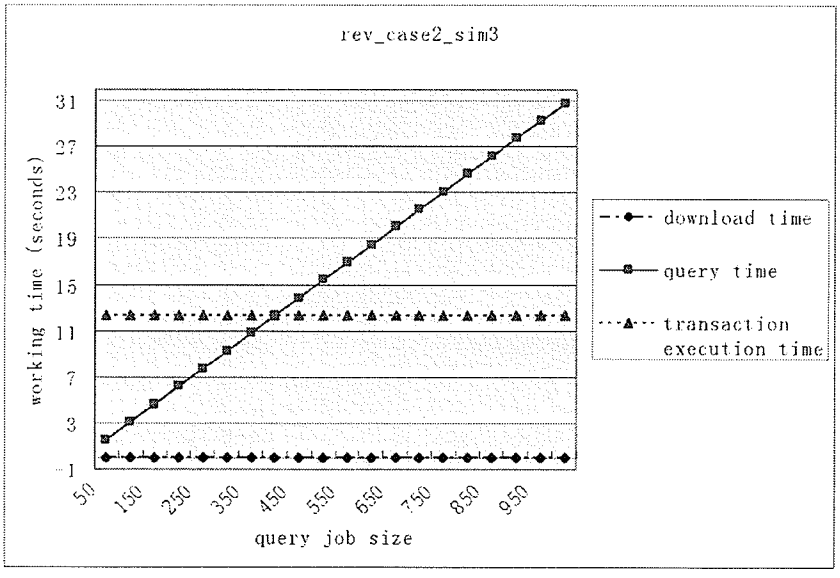


(b) Results from solution of the model.

Figure 5.17: Simulation of the REV paradigm model with handoff when varying the transaction message number.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the code to be sent	code_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.18: Simulation of the REV paradigm model with handoff when varying the query job size.

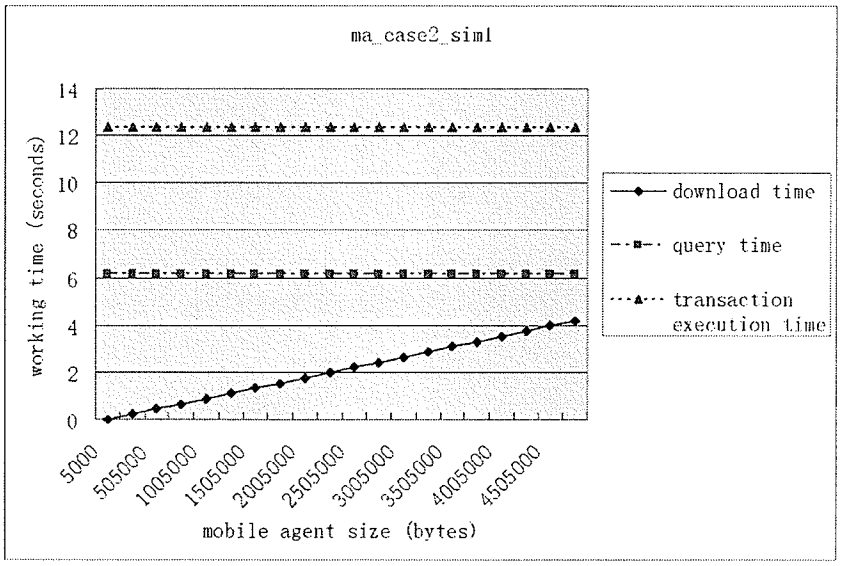
of the performance variables in the simulation of the system with handoff plugged with the MA paradigm when varying the size of the mobile agent generated and sent by the mobile unit to the server. The same range is defined for the mobile agent size as the range for the code size in the COD and REV paradigms for simplifying the application. The results from solution of the model of the system are indicated in Figure 5.19(b) which shows the download time increases from 4 milliseconds to 4 seconds with the increments of the mobile agent size, the query time is about 6 seconds when the server processes 200 query jobs, and the transaction execution time is about 12 seconds when the mobile unit sends 8 transaction messages.

Figure 5.20(a) displays a group of parameter values employed in the simulation of the system with handoff plugged with the MA paradigm when varying the number of transaction messages sent by the mobile unit to the server. In the system the mobile unit sends the mobile agent, the transaction messages and the location notification message; thus, the transaction message number should be the result of subtracting 2 from the number of the messages to be sent by the mobile unit. The results from solution of the model of the system are illustrated in Figure 5.20(b) which reveals the transaction execution time increases from 1 second to 148 seconds with the increments of the transaction message number from 1 to 96, the download time is 44 milliseconds when the mobile agent size is defined as 50000 bytes, and the query time is about 6 seconds when the query job size is defined as 200 and the query message size is defined as 1000 bytes.

Figure 5.21(a) represents a batch of parameter values used for calculating the results of the performance variables in the simulation of the system with handoff plugged with the MA paradigm when varying the size of the query jobs processed by the server after it has received the query message. The results from solution of the system model are shown in Figure 5.21(b) which indicates the query time of the system is going up from 1 second to 30 seconds with the increments of the query job size, the download time is equal to the download time of the same system when varying the transaction message number, and the transaction execution time is equal to the transaction execution time of the same system

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	[5000.0-4755000.0] [Incremental Range,250000.0]
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.

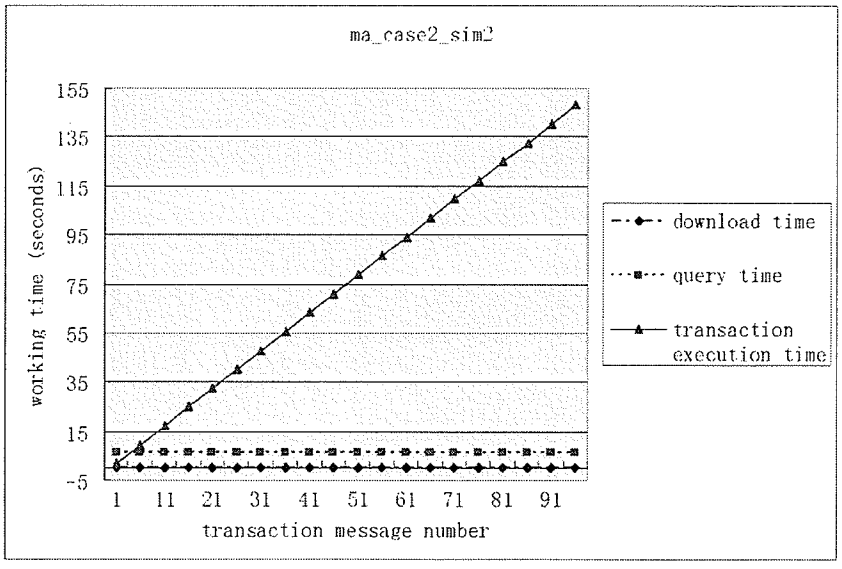


(b) Results from solution of the model.

Figure 5.19: Simulation of the MA paradigm model with handoff when varying the mobile agent size.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	200.0
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	[3-98] [Incremental Range,5]
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.



(b) Results from solution of the model.

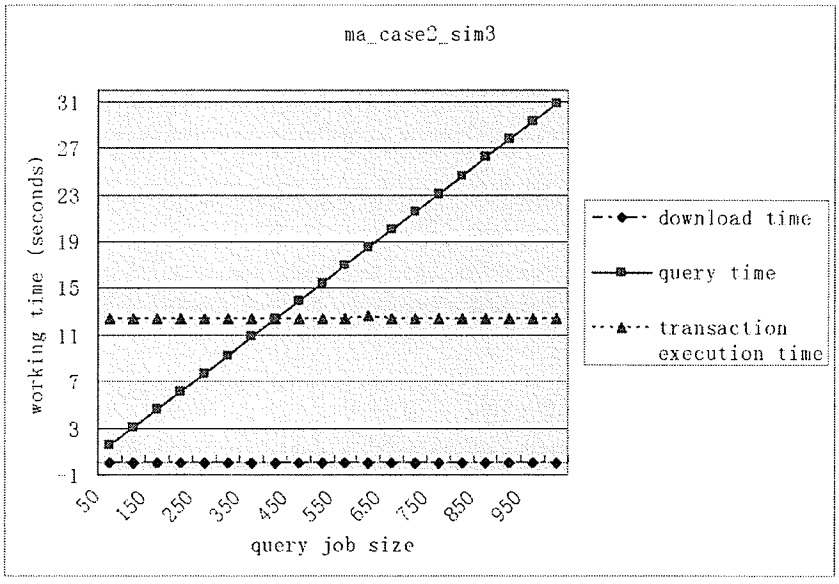
Figure 5.20: Simulation of the MA paradigm model with handoff when varying the transaction message number.

when varying the mobile agent size.

Figure 5.22 represents the comparison of the total working time of the systems with handoff plugged with different code mobility paradigms when varying the size of the code or the mobile agent, and it reveals the MA paradigm has the worst performance and the other two paradigms has very similar performance. However, the working time of the system plugged with the COD paradigm is 5 milliseconds faster than the REV paradigm. Figure 5.23 displays the comparison of the total working time of the systems with handoff plugged with different code mobility paradigms when varying the transaction message number, and it shows the differences between the performance of three code mobility paradigms can be ignored. The data show that the REV paradigm is 13 milliseconds faster than the MA paradigm and the COD paradigm is 5 milliseconds faster than the REV paradigm. Figure 5.24 represents the comparison of the total working time of the systems with handoff plugged with different code mobility paradigms when varying the query job size from which three code mobility paradigms are shown to have similar performance. However, with the rise of the query job size the COD paradigm shows the continuously increscent superiority that is of 0 to 25 milliseconds than the REV paradigm and the REV paradigm represents the continuously increscent superiority that is of 7 to 51 milliseconds than the MA paradigm.

Parameter	Variable	Value
Probability of transaction abortion	abort_prob	0.05
Size of the mobile agent	ma_size	50000.0
Size of the query message	qmsg_size	1000.0
Size of the query jobs to be processed by the server	qjob_size	[50.0-1000.0] [Incremental Range,50.0]
Size of the flights info to be returned	qresults_size	8000.0
CPU rate of the server	scpu_rate	1946.0
Size of the transaction message	txmsg_size	500.0
Size of the transaction jobs to be processed by the server	txjob_size	50.0
Number of the messages to be sent by the mobile unit	mu_msgnum	10
Size of the log message	logmsg_size	500.0
Size of the confirmation message	cfmmsg_size	50.0
Wireless link speed	wireless_speed	1250000.0
Wired link speed	wired_speed	1.25E7
Size of the admission information	adminfo_size	500.0
Size of the transaction control message	ctrltx_size	500.0
Size of the handoff request message	hdfreq_size	500.0
Size of the mu's location information message	locinfo_size	1000.0
Probability of handoff request rejection	rej_prob	0.1
Size of the resource to be transferred between two base stations	res_size	200000.0

(a) Parameter values for the simulation.



(b) Results from solution of the model.

Figure 5.21: Simulation of the MA paradigm model with handoff when varying the query job size.

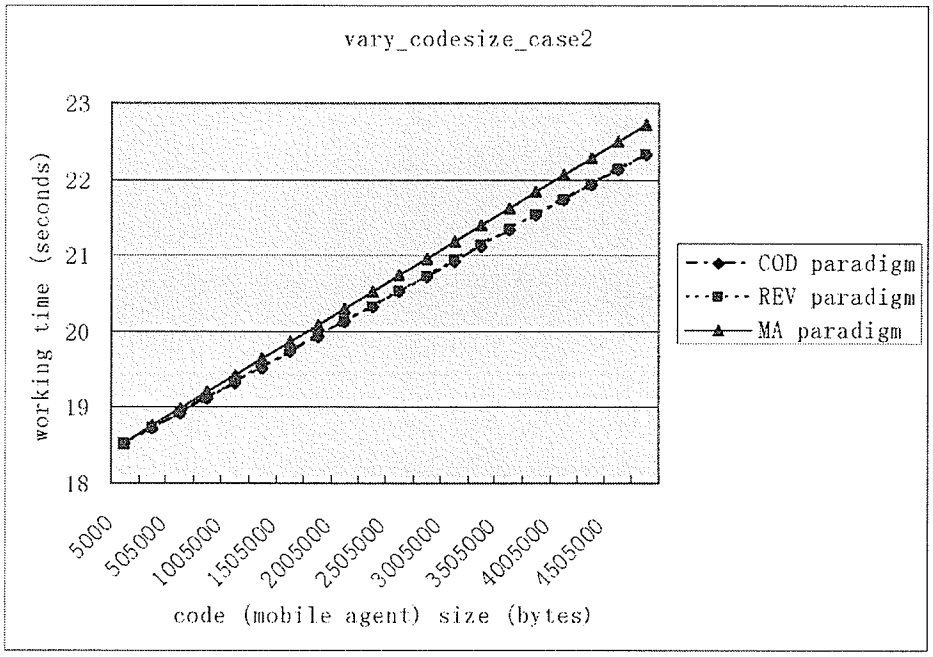


Figure 5.22: Comparison of results when varying the code size.

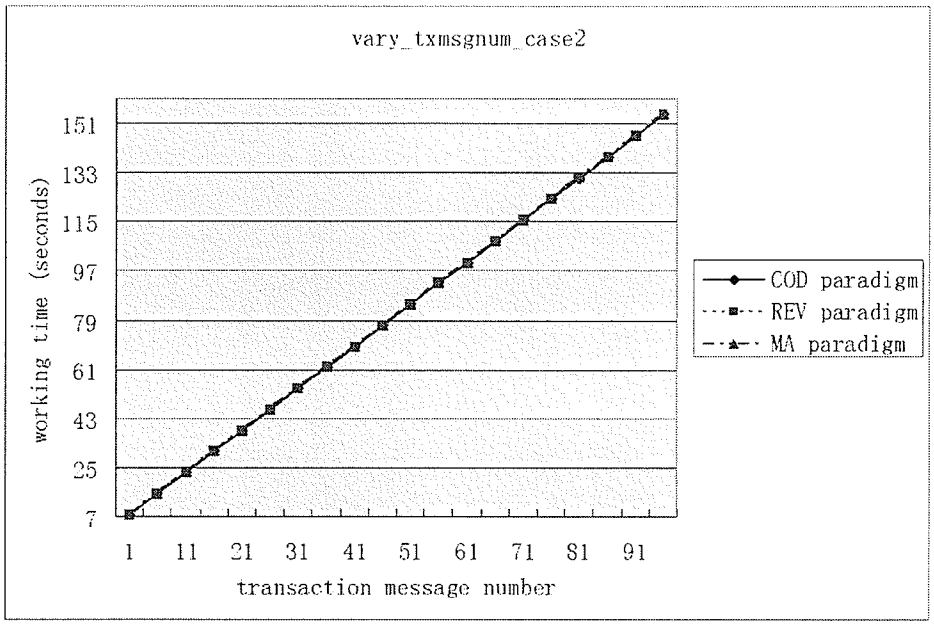


Figure 5.23: Comparison of results when varying the transaction message number.

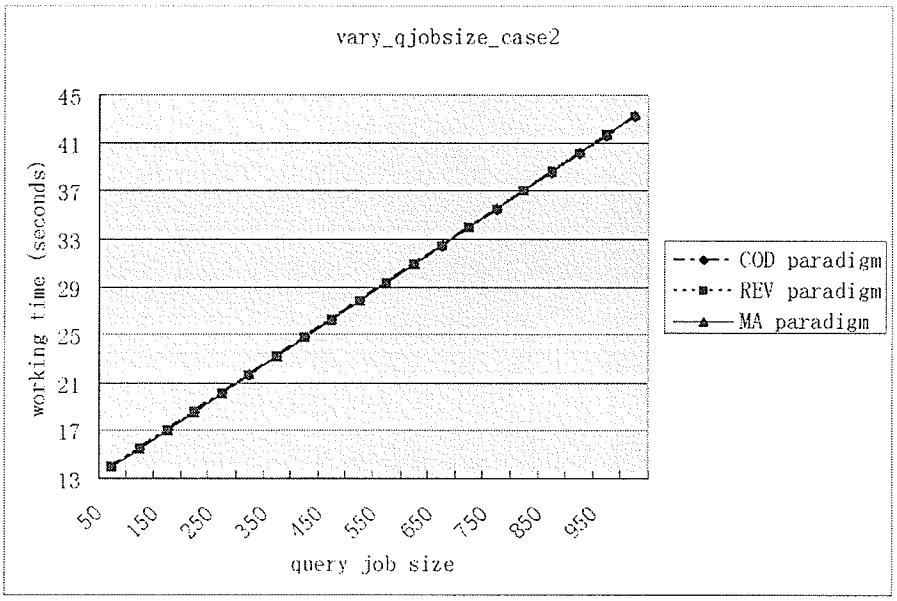


Figure 5.24: Comparison of results when varying the query job size.

Chapter 6

Conclusions

Modern software systems often operate in large scale distributed, even mobile environments. To effectively design complex software systems, the design process should be accompanied by quantitative evaluation of different design alternatives. Quantitative analysis of software systems has been recognized to be important and useful for requirements and design, and, specifically, performance analysis should be integrated in the software development life cycle from the early stages. Such quantitative evaluation considers measures like response times, queue lengths, throughput, or fail probabilities and helps understanding system performance. To this aim, several approaches have been proposed to integrate or combine performance analysis and software architecture specification. The technologies of mobile computers and wireless networks result in emerging of a new computing paradigm, mobile computing which will affect the design of much of the current systems software, including that of database systems. Although many methodologies have been developed for modeling and assessing the performance of software systems, few efforts have been made on the research of mobile software systems presenting both physical and code mobility.

In the thesis, an approach has been proposed for generating the performance models of mobile software systems by converting the UML Sequence Diagrams of systems into Stochastic Activity Networks and comparing the performance of different design alternatives based on the results of solving those models. First we presented the necessity and

requirements for developing the methods of integrating the performance evaluation with the system modeling on the early stage of the mobile software design, with the features of mobile software systems introduced including the physical device mobility, the code mobility paradigms probably adopted in the software application, and the transaction mobility presented to the mobile database systems through a scenario. We reviewed the existing work on the model-based performance prediction in software development and emphasized on the contribution and drawbacks of the methodologies for modeling and performance analysis of mobile software systems. A mobile software system is characterized by several aspects such as the structure and the behavior of the software systems, the mobility of the software modules, and the mobility of the devices accessing the system which all are considered in our approach of performance modeling. The referred scenario is simplified as an example of this modeling approach which adopts the UML Sequence Diagrams (SDs) to describe the behavior of the system and the Stochastic Activity Networks (SANs) to model the system including its different aspects of mobility. Even though code mobility is regarded as an application level design paradigm and many researchers suggest that a major benefit provided by mobile code is the capability to reduce network communication by moving client's knowledge close to server's resources, thus accessing them locally, these arguments need to be supported by a careful analysis of the problem, possibly supported by quantitative evidence. Hence, three code mobility paradigms are inserted in the pre-existing system model respectively, generating the performance model which is solved by the Mobius tool to get the final quantitative evaluation results. This developed methodology is summarized as follows,

- Model the system behavior using UML Sequence Diagrams
- Derive SAN models of systems from UML SDs
- Derive SAN models of code mobility diagrams from UML SDs
- Plug SAN models of code mobility diagrams into pre-existing system models and generate final performance models

- Execute the simulations of performance models
- Evaluate the results generated from simulations

In this work, a mobile software system in which a database is residing on the server accessed by the mobile unit, possibly through different base station is introduced as the example for the experiment of the approach, and all queries and transactions are assumed to be processed by the server, which means the transaction and query messages have to be forwarded to the server ultimately. Moreover, two cases that the mobile unit stays in one cell and it moves to another cell are considered for modeling physical mobility. Three code mobility are assessed quantitatively without comparing with the traditional client-server interaction paradigm; however, the performance of the client-server paradigm can be easily evaluated by solving the pre-existing system model without inserting any code mobility paradigm. This methodology fills the gap between system design and performance evaluation of the mobile software systems by translating the UML sequence diagrams into SAN performance models. The future work includes the extension of the methodology to complete the algorithm of the translation from the UML SDs to SAN models and the validation of the methodology on real world industrial case studies.

Bibliography

[1] M. Ajmone, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press, Cambridge, Mass., 1986.

[2] G. Balbo. Introducation to stochastic petri nets. *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*, pages 84–155, 2002.

[3] M. Baldi and G. P. Picco. Evaluating the tradeoffs of mobile code design paradigms in network management applications. In *Proceedings of the 20th International Conference on Software Engineering*, pages 146–155, Kyoto, Japan, 1998.

[4] S. Balsamo and M. Marzolla. Towards performance evaluation of mobile systems in uml, 2003.

[5] L. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation.* Prentice Hall, 2000.

[6] S. Bernardi, S. Donatelli, and J. Merseguer. From uml sequence diagrams and statecharts to analysable petri net models. In *Proceedings of the 3rd international workshop on Software and performance*, pages 35–45, Rome, Italy, 2002.

[7] M. Bernardo, P. Ciancarini, and L. Donatiello. Æmpa: a process algebraic description language for the performance analysis of software architectures. In *WOSP '00: Proceedings of the 2nd international workshop on Software and performance*, pages 1–11, New York, NY, USA, 2000. ACM Press.

[8] P. Bracchi and V. Cortellessa. A framework to model and analyze the performability of mobile software systems. In *Proceedings of the 4th International Workshop on Software and Performance*, volume 29 of *1*, 2004. ACM SIGSOFT Software Engineering Notes.

[9] A. Carzaniga, G. P. Picco, and G. Vigna. Designing distributed applications with mobile code paradigms. In *Proceedings of the 19th International Conference on Software Engineering*, pages 22–32, Boston, MA, 1997.

[10] V. Cortellessa and V. Grassi. A performance-based methodology to early evaluate the effectiveness of mobile software architectures. *Journal of Logic and Algebraic Programming*, 51(1):77–100, April 2002.

[11] V. Cortellessa and R. Mirandola. Deriving a queueing network based performance model from uml diagrams. In *Proceedings of the 2nd international workshop on Software and performance*, pages 58–70, Ottawa, Ontario, Canada, 2000. ACM Press.

[12] D. Daly and et al. Mobius: An extensible framework for performance and dependability modeling. In *Proceedings of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99)*, Zaragoza, Spain, Sept. 1999.

[13] R. A. Dirckze and L. Gruenwald. A pre-serialization transaction management technique for mobile multi-databases. *ACM Mobile Networks and Applications*, 5(4):311–321, 2000.

[14] OMG Documentation. *Unified Modeling Language (UML),version 2.0.* http://www.omg.org/technology/documents/formal/uml.htm, 2005.

[15] M. H. Dunham, A. Helal, and S. Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *ACM-Baltzer Journal on Mobile Networks and Applications (MONET)*, 2(2):149–161, 1997.

[16] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, Sept. 1998.

[17] S. Gilmore and J. Hillston. The PEPA workbench: A tool to support a process algebra-based approach to performance modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in Lecture Notes in Computer Science, pages 353–368, Vienna, may 1994. Springer-Verlag.

[18] V. Grassi and V. Cortellessa. Performance evaluation of mobility-based software architectures. In *Proceedings of the 2nd international workshop on Software and performance*, pages 44–46, Ottawa, Ontario, Canada, 2000.

[19] V. Grassi and R. Mirandola. Uml modelling and performance analysis of mobile software architectures. In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, pages 209–224, London, UK, 2001. Springer-Verlag.

[20] V. Grassi and R. Mirandola. Primamob-uml: a methodology for performance analysis of mobile software architectures. In *Proceedings of the 3rd International Workshop on Software and Performance*, pages 262–274, Rome, Italy, 2002.

[21] V. Grassi, R. Mirandola, and A. Sabetta. Uml based modeling and performance analysis of mobile systems. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 95–104, Venice, Italy, 2004. ACM Press.

[22] A. Helal and M. Eich. Supporting mobile transaction processing in database systems. Technical Report 95–003, TRCSE, University of Texas, Arlington, 1995.

[23] H. Hermanns, U. Herzog, and J. P. Katoen. Process algebra for performance evaluation. *Theoretical Computer Science*, 274(1–2):43–87, 2002.

[24] H. Hermanns, U. Herzog, U. Klehmet, V. Mertsiotakis, and M. Siegle. Compositional performance modelling with the tipptool. *Performance Evaluation*, 39(1–4):5–35, 2000.

[25] L. Ismail and D. Hagimont. A performance evaluation of the mobile agent paradigm. In *Conference on Object-Oriented*, pages 306–313, 1999.

[26] K. Kant. *Introduction to Computer System Performance Evaluation.* McGraw-Hill College, 1992.

[27] P. King and R. Pooley. Derivation of petri net performance models from UML specifications of communications software. In *LNCS 1786: Computer Performance Evaluation, Modelling Techniques and Tools— 11th International Conference, TOOLS 2000, Schaumburg, IL, USA, March 2000. Proceedings / Boudewijn R. Haverkort, Henrik C. Bohnenkamp, Connie U. Smith (Eds.)*, pages 262–276. Springer Verlag, 2000. InternalNote: hr.

[28] D. A. Menascé. A framework for software performance engineering of client/server systems. In *Proceedings of 1997 Computer Measurement Group Conference*, Dec. 1997.

[29] D. A. Menascé and H. Gomaa. A method for design and performance modeling of client/server systems. *IEEE Transactions on Software Engineering*, 26(11):1066–1085, Nov. 2000.

[30] J. Merseguer, J. Campos, and E. Mena. Evaluating performance on mobile agents software design. *Actas de las VIII Jornadas de Concurrencia*, pages 291–307, Jun 2000.

[31] J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: structure, behavior and application. In *Proceedings of International Workshop on Timed Petri Nets*, pages 106–115, Torino, Italy, 1985.

[32] D. Milojičić, F. Douglis, and R. Wheeler, editors. *Mobility: processes, computers, and agents*, chapter 1. Addison Wesley Longman, 1999.

[33] M. K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, C-31(9):913–917, Sept. 1982.

[34] A. Movaghar. Stochastic activity networks: A new definition. In *Proceedings of the IASTED International Conference on Modeling and Simulation (MS'97)*, pages 27–30, Pittsburgh, PA, 1997.

[35] T. Murata. Petri nets: properties, analysis and applications. In *Proceedings of the IEEE*, volume 77 of *4*, pages 541–580, 1989.

[36] Object Management Group. *CORBA: Architecture and Specification*, August 1995.

[37] Telecommunication Standardization Sector of ITU. *Message Sequence Charts.* ITU-T Recommendation Z.120, 1998.

[38] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[39] E. Pitoura and B. Bhargava. Revising transaction concepts for mobile computing. In *Proceedings of the IEEE Workshop on Mobile Systems and Applications*, pages 164–167, Santa Cruz, CA, USA, 1994.

[40] E. Pitoura and B. Bhargava. Maintaining consistency of data in mobile distributed environments. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, pages 404–413, 30 May–2 June 1995.

[41] R. Pooley. Using uml to derive stochastic process algebra models. In *Proceedings of the 15th UK Performance Engineering Workshop*, pages 23–33, Bristol, July 1999.

[42] M. L. Puterman. *Markov Decision Processes.* John Wiley and Sons, Inc., New York, NY, USA, 1994.

[43] C. U. Smith and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software.* Addison Wesley, 2002.

[44] B. Venners. Under the hood: The architecture of aglets. JavaWorld: IDG's magazine for the Java community, 2(4), Apr. 1997.

[45] G. D. Walborn and P. K. Chrysanthis. Transaction processing in pro-motion. In *ACM Symposium on Applied Computing*, pages 389–398, San Antonio, Texas, USA, 1999.

[46] M. Wermelinger and J. L. Fiadeiro. Connectors for mobile programs. *IEEE Transactions on Software Engineering*, 24(5):331–341, May 1998.

[47] J. E. White. Telescript technology:the foundation of the electronic marketplace. General Magic Inc., 1994.

[48] L. G. Williams and C. U. Smith. Performance evaluation of software architectures. In *Proceedings of the 1st international workshop on Software and performance*, pages 164–177, Santa Fe, New Mexico, United States, 1998. ACM Press.