Mobility Information and Mobile Transaction Processing

BY

Jian CHEN

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements for the degree of

MASTER OF SCIENCE

Department of Computer Science

University of Manitoba

Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

MOBILITY INFORMATION AND MOBILE TRANSACTION PROCESSING

BY

JIAN CHEN

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

MASTER OF SCIENCE

Abstract

In a mobile environment, system and application performance is impacted by mobility factors such as varying bandwidth, frequent disconnection, and changing locations. These factors make mobility information more important in the design of distributed algorithms. However, at present, no general model exists for exploiting the availability of mobility information. Focusing on this issue, this thesis investigates mobility environment modelling; especially the classification, representation, and utilisation of mobility factors.

Based on the concepts and methods of modelling mobility factors proposed in this research, an adaptive transaction scheduling (ATS) approach for mobile transaction processing, chosen as a particular application domain, is developed to demonstrate the feasibility of designing better distributed applications by making use of mobility information. The ATS approach is designed to support autonomous operations during network disconnection and dynamically select the site of transaction execution based on mobility information such as bandwidth and disconnection ratio.

The performance of the ATS approach is evaluated against fixed transaction processing policies by the means of simulation experiments. A queue simulation model is designed and implemented in discrete-event simulation language. The results of the simulation experiments show that adaptive transaction processing produces better performance and confirms that the design of applications in mobile environments can benefit from available mobility information.

Acknowledgements

I deeply thank my supervisor, Dr. Randal Peters, for his support, encouragement and patience during the course of my graduate study.

I would like to thank my thesis committee members for their time, valuable suggestions and comments.

I have benefited a lot from the courses taught by Dr. N. Arnason, Dr. Peter Graham and Dr. Randal Peters, and from the academic discussions of the database research group in the department led by Dr. Ken Barker.

I would also like to thank my fellow students for their companionship.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This chapter describes general concepts and issues of mobile computing, and presents the research hypothesis, goals and problems covered by this thesis. The major work performed in this thesis is summarised as well.

1.1 Mobile Computing

Mobile computing is a computing paradigm and is perceived as a natural evolution of the wired computing environment. A wired network brings isolated computers together, and provides users with resource-sharing and communication capability from pre-defined locations. In a mobile computing environment (MCE), with a wireless network, the fixed location assumption is removed, and end-users, freed from physical connection to the underlying network, are allowed to access information and share resources from anywhere, at any time.

The integration of personal computing with wireless networks will create an entirely new class of applications. The applications in the mobile environment can be generally distinguished into two categories: data transmission applications and data management applications. The examples of data transmission applications include sending electronic mail, location-dependent or time-sensitive information broadcasting, and document exchange. Data management applications access shared data resources that require maintenance of their consistency. These kinds of applications

have a large potential in some business areas that require field data access and co-operative design. The requirements of field data access come from positions such as automobile insurance salesperson [MHB97], marketing officer (e.g., a wheat marketing officer [YZ94b, YZ94a], a financial investment manager), or field worker (e.g., a GIS scientist [DBCF94]). Examples of co-operative design include co-operative document authoring, co-operative calendar managers, and co-operative meeting schedulers [KTW97a, KTW97b].

The design of conventional distributed algorithms is based on a static model where a set of processes run on stationary machines and communicate by means of message sending over point-to-point logical channels. The hosts at the endpoints of the channel do not change with time. The network communication is considered reliable and a network partition is treated as a failure. Furthermore, network stations are considered to have sufficient computing resources and can communicate with the rest of the system with equal capability.

Mobile computing is distinguished from the conventional distributed computing in several aspects. First, the connection of mobile computers to the rest of network is by wireless links. Compared to a wired network, wireless links are relatively unreliable and have much lower bandwidth. Second, mobile computers may connect from different locations while retaining their connection to the network. This is known as *roaming* and causes the network system to dynamically re-configure communication so that messages are routed smoothly and without an interruption in service as a mobile unit changes location. Third, mobile computers suffer from limited computing resources such as short battery life, small size of display, and small volume of storage. All these factors make the computing environment much more uncertain and in turn raise new technical issues and problems that must be solved.

1.2 Motivation

The challenges of mobile computing stem from the dynamics of the networking environment, the flexibility of user movements, and the limitation of system resources.

Although these factors make a computing system more complicated, some opportunities for resource sharing, information access, and message routing are created in designing better distributed algorithms.

This research is motivated to capture mobility information in a mobile environment to support the design and application of adaptive approaches in the data management area with an emphasis on transaction processing. The hypothesis in this research is that better performance of transaction processing in a mobile environment can be achieved if some information concerning the current environment status of a mobile host is available. Therefore, the goal is to explore the mechanisms that can improve the performance of transaction processing in a mobile environment.

An *adaptive approach* is a strategy that allows a computing system to perform certain actions based on its perception of the current system environment. For example, to deal with mobility, some techniques can be used to improve the quality of service in multimedia applications by capturing environment information [NPS95, ABB+96]. Suppose an application wishes to view on-line images. If the bandwidth of the current communication network is diminishing, the application may switch from full-colour images to black and white images. Meanwhile, a different algorithm for image compression can be applied.

Adaptive strategies have drawn the attention of researchers in mobile computing because mobility implies dynamics and requires adaptation [Kat94]. Adaptive approaches have been proposed under different names from various perspectives in the literature. These include *application-aware adaptation* [NPS95] , *context-aware applications* [SAW94], *adaptive services* [DBCF94] and, *adaptive protocols* [JM96].

The performance of transaction processing in mobile computing is impacted by varying communication conditions as well as available computing resources. We cannot assume the existence of a uniform computing environment for all the individual mobile hosts because each mobile host can have its own mobile behaviour and computing condition. The computing condition of a mobile host can be reflected by various mobility factors, such as *bandwidth*, *disconnection frequency*, *battery life*, and

movement pattern. Such mobility factors are of the utmost importance in the design of a distributed algorithm. Furthermore, they are generally time and location dependent.

In a mobile environment, it is typical for a mobile host to frequently disconnect from the rest of the network and encounter varying communication bandwidths as it moves to different locations. The disconnection in a mobile environment can be distinguished into *predictive* and *non-predictive.* A predictive disconnection is performed deliberately by a user in some circumstances to reduce energy and/or communication costs. An explicit disconnection/reconnection protocol can be applied to handle a predictive disconnection. A non-predictive disconnection can happen in several circumstances. For example, when a mobile host moves out of a service area of wireless communication without prior announcement or the computer enters suspend mode because of a low battery condition. Therefore, non-predictive disconnection occurs without prior intention by the user.

Frequent disconnection of a mobile computer resulting from low bandwidth, short battery life, and occasional movement out of a service area can deteriorate the performance of transaction processing. Hence, it has been identified as one of the significant issues in mobile distributed database management [AK93, Duc92, MDC93]. The approach in this thesis attempts to deal with the issue caused by non-predictive disconnection.

1.3 Research Issues

Most commercial distributed database systems are designed to support the client-server computing mode. The execution of transactions submitted by a client is always co-ordinated by a database server. Transaction processing algorithms in a conventional distributed environment assume that the network environment is reliable. Thus, the disconnection of a client is treated as a failure and handled by a recovery procedure. Therefore, traditional transaction management does not have to consider the operations performed by a mobile host during its time of disconnec-

tion. Moreover, due to network stability and adequate computing resources, it is not necessary to monitor connectivity conditions and change the processing procedures dynamically.

In mobile computing, the mobility of end-users implies that a mobile user does not have to submit the operations of a transaction to distributed data servers from a fixed location. The limited local computing resources on a mobile host, especially short battery life and its flexible movement capability, allow it to disconnect from time to time. In such a dynamic environment, it poses new challenges to increase concurrency, reduce the transmission of messages among the hosts, maintain the consistency of the database, and balance the utilisation of computing resources.

The intention of this research is to develop an adaptive approach to support transaction processing in a mobile environment. This requires a dynamic selection of an optimal action from a specified set of alternatives. Generally, the execution of a transaction follows one of three policies in a conventional distributed database environment. These are known as : (i) *data request shipping*, (ii) *function request shipping*, and (iii) *I/O request shipping* [JKT88, Tho96]. Each policy has its own merits under appropriate computing conditions.

The communication bandwidth in a conventional distributed environment is assumed to be constant and highly reliable. Therefore, the performance of a transaction execution policy in terms of response time is a proportion to the data volume of transmission. However, in a mobile environment, due to varying network conditions and host movement, this is no longer the case. It is necessary to exploit the advantages of different data accessing policies under specified application objectives and given computing conditions. This suggests that mobile transaction processing should not be restricted to a fixed policy. Instead, dynamic decision-making should be applied.

Selecting an optimal action dynamically requires an evaluation of the potential benefit for a given policy according to the current observable computing environment. Consequently, the knowledge of the environment status is required. If information about a mobile environment can be captured, a proper strategy of transaction pro-

cessing can be applied so that the overall performance of data management can be improved. To reach this goal, the following two specific issues are investigated in this thesis:

1. **Modelling of the mobile environment:**

   (1) investigate the environment features essential to the design of effective adaptive approaches;

   (2) explore the representation and measurement of mobility information;

2. **Application of mobility information in transaction processing:**

   (1) design a transaction processing model that can take into account mobility information;

   (2) study the impact of mobility information on transaction processing;

1.4 Overview of Thesis

The work in this thesis does not focus on technique implementation, but rather on concepts and theoretical ideas. Following the research hypothesis that an adaptive approach can improve performance of mobile computing, this thesis presents the research results on modelling mobility and the application of mobility information in transaction processing.

First, the design space of mobile networking systems is investigated. The structure complexity as well as environment complexity of potential mobile computing systems are addressed. Second, to describe environment complexity of a mobile system, the issue of modelling environment mobility is studied. The concepts and measuring methods concerning the mobility of a mobile host, i.e. *unit mobility* is proposed. The unit mobility as a general-purpose model is intended to combine multiple mobile factors together and offers more accurate information to support the better design of distributed applications in mobile computing. Third, to demonstrate

the utilisation of mobility information, a particular application model, i.e. *Adaptive Transaction Scheduling* (ATS) in data management area, has been developed. The design of the ATS approach explores the availability of mobility information and aims at improving the performance of transaction processing with the existence of network disconnection. Finally, to evaluate the impact of utilising mobility information on the performance of transaction processing in a mobile environment, a simulation model has been created and implemented. The performance of the ATS approach has been compared with two other approaches, i.e. fixed data shipping approach and fixed transaction shipping approach. The results of simulation experiments show that the ATS approach in general produces better performance in terms of overall response time, elapsed processing time on a mobile host, and total number of transaction throughputs. This confirms the hypothesis of the thesis research.

1.5 Organisation of Thesis

The remainder of the thesis is organised as follows. Chapter 2 discusses the issue of modelling the mobile environment and stresses the importance of capturing mobility information. In this chapter, the classification, representation and utilisation of mobility factors are investigated. The concepts of unit mobility and its measurement are defined.

Chapter 3 focuses on the development of a transaction processing model. In this chapter, the limitation of existing transaction models and the characteristics of mobile transaction processing are discussed. After identifying possible transaction processing policies, an adaptive transaction scheduling approach is proposed. The approach allows a mobile transaction to be locally committed when temporary disconnection occurs and also dynamically selects the transaction processing site. It serves as a particular application model to demonstrate the benefit of available mobility information.

To evaluate the performance of adaptive transaction scheduling approach, a queue simulation model is developed. The design of a simulation model is described in

Chapter 4. The methods and results of simulation experiments are reported in this chapter as well.

Chapter 5 reviews the related work. As the research in this thesis falls into the category of adaptive approaches, existing adaptive approaches at different system levels are described. Focusing on the data management area, typical mobile transaction processing approaches appearing in current literature are summarised. Finally, conclusions and future work are presented in Chapter 6.

Chapter 2

MODELLING MOBILITY

In a mobile environment, system and application performance is impacted by mobility factors. The mobility information is of the utmost importance in the design of a distributed algorithm. However, at present, no general model exists for exploiting the nature of mobility. In this chapter, from a system viewpoint, the issue concerning modelling mobility with a focus on the classification, representation and utilisation of mobility information is investigated.

2.1 A Classification of Mobile Computing Systems

At present, mobile computing has not had a commonly accepted system model. To prepare the groundwork for this research, the chapter begins with a classification of potential mobile computing systems.

A mobile computing system can be viewed as an extension of a traditional distributed system, characterised by two types of system architectural components: mobile computers and wireless communication facilities. A mobile computer, also termed an *mobile unit*, equipped with a wireless communication interface can maintain network connection while moving. It is easy to understand that with more mobile computers in a system, the resource control and management become more difficult. According to its role, a mobile computer can be further distinguished into mobile

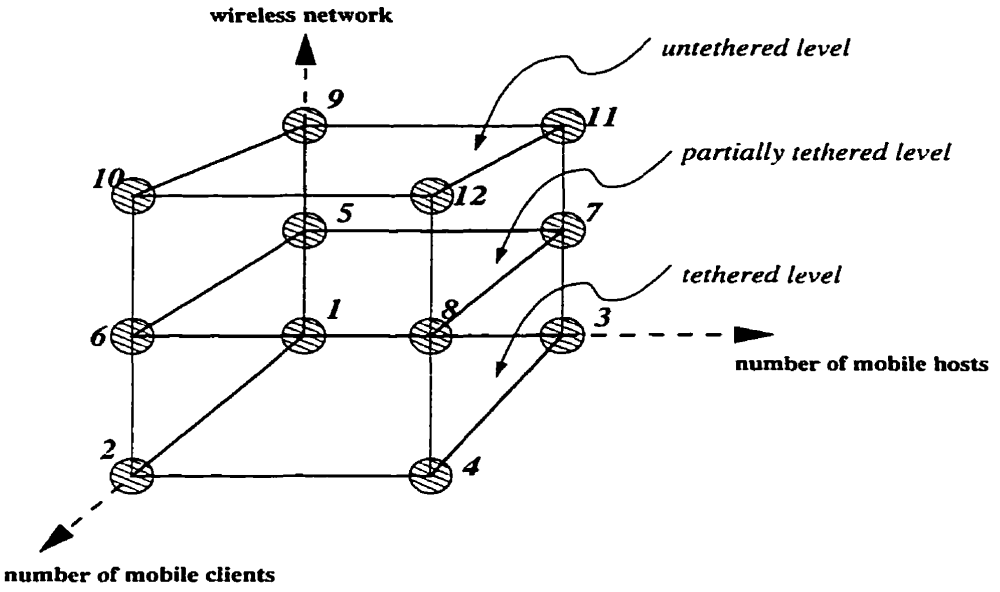


Figure 2.1: A classification of mobile network systems

client or mobile server. A *mobile client* is a computer that changes its locations, but only plays a role of resource consumer and provides no service for others. Conversely, a *mobile server* or *mobile host* is a computer that changes its locations as well as provides system services or shared resources.

Wireless communication facilities establish the network connection between computers in a mobile computing system. Without wireless communication, the topology of network is stable and the mobility of computers in a system is limited to location changes at discrete time points. However, with the introduction of wireless communication facilities, the mobility of computers goes from discrete time feature to continuous time feature. That is, the location change of a mobile computer can take place at any moment. This certainly increases the flexibility of end-user movement.

From the viewpoint of system architecture, it can be claimed that the structure complexity of a mobile computing system is dependent on the proportion of wireless communication, the number of mobile clients, and the number of mobile servers in the system. Thus, the design space of a mobile computing system is constrained by these factors. A mobile computing system can be in any position of its design space as illustrated by Figure 2.1.

In the design space there are three specific surfaces corresponding to three levels of complexity of mobile computing systems. They are generally named as *tethered mobile system level*, *partially tethered mobile system level*, and *untethered mobile system level*.

At a tethered mobile system level, all the computers are networked by physical lines, but they may change their locations. For instance, laptops can be transported and connected to the network at different locations. Some work stations may also be relocated, but this is not likely because of the physical characteristics. It can be seen that a traditional distributed system is just a special case of tethered mobile system, in which both the number of mobile clients and the number of mobile servers are zero. In other words, all the computers are fixed. A fixed computer is called a *fixed unit* or a *fixed host*. It does not change its location and is always connected by physical lines.

A mobile computing system at the partially tethered level means that some computers are connected by the wireless network and some are connected by the wired network. In other words, a system includes both wired and wireless networks. Due to wireless networking, retaining communication while moving is feasible. The simplest case of a partially tethered computing system is that no computers are moving, even though some are using wireless communication. Its most complicated case is when both clients and servers are allowed to move to different locations.

The untethered mobile system level denotes that all the computers in the system are wirelessly networked, or, there is no fixed network. At this level, similarly, its simplest case is that no computers change locations. The mobility only reflects the uncertainty of wireless network connections, such as varying bandwidths and disconnection/reconnections. If all the computers are allowed to change their locations, it becomes the most complicated case in mobile computing. This is usually known as an *ad hoc* mobile system in the current literature [JM96].

The classification presented here describes a general design space of a mobile computing system and provides a more concrete description about a given system.

In fact, relative to the environment complexity of a mobile computer (captured by the unit mobility), structure complexity of a mobile computing system can be viewed as the mobility at the system level, or system mobility. Higher system mobility implies a more unstable network topology, more varying system resource status, and higher user flexibility. As we can see from the classification, such system mobility increases from tethered system level to untethered system level. At a given level, the mobility changes from point to point. For instance, the complexity of a system at vertex 6 is greater than vertex 7 because the former one only involves mobile clients and all the hosts are fixed. Similarly, with the classification, an *ad hoc* system can be further distinguished into more specific cases, like the systems at the vertices 10, 11 and 12 in Figure 2.1. In current literature, they are generally treated as one type of system.

2.2 Introduction to Unit Mobility

The classification of mobile network systems above offers general information about the structure complexity of mobile computing systems. However, it does not reveal the environment complexity faced by individual computers within a given system. This issue is addressed in the rest of the chapter.

The environment complexity of a mobile computer roughly reflects the degree of instability of its external computing resources associated with other computers. The degree of instability of computing resources of a mobile computer is referred to as the mobility of the mobile unit, or *unit mobility* for short (a formal definition is given in section 2.5).

The unit mobility is impacted by the mobile behaviour of a computer (e.g., physical movements, disconnection) as well as the status of available computing resources (e.g., communication bandwidth, residual battery life). Conversely, the changes of unit mobility describe the availability of its resources for the other systems in a current environment. It would be more beneficial to the design and application of mobile systems if unit mobility information is available. For example, if a mobile unit disconnects frequently, it will be difficult for other computers to share the data

on the mobile unit. Thus, a different replication scheme may be required.

Researchers in the mobile computing area are aware of the importance of capturing mobility information of mobile units. The efforts have been made from different research dimensions. For example, the work in [Lin94, LM95] pays attention to location changes. It is believed that if the next location where a mobile unit moves to can be predicted, required computing resources can be scheduled in advance and better system services can be provided. The work reported in [MES95] models communication resources. Focusing on network bandwidth, the network connectivity of a mobile unit is distinguished from three status identifiers: strongly-connected, weakly-connected, or disconnected. An appropriate system reaction can be taken, say, changing the time-out parameter for a remote system call, if the status of network connectivity of a mobile unit is detected.

Information captured based on a particular factor is useful, but inadequate in terms of modelling mobility. Mobility is attributed to multiple factors. Information derived from a factor is just from one dimension. For instance, given two mobile units that have the same network bandwidth, it is more difficult to access the resources on a frequently-moving unit rather than a stable one. Thus, they display different mobility. Similarly, a frequently disconnected unit is more mobile than a constantly connected one. This suggests that it would be necessary to explore an integrated scheme to reflect the general mobile status of a mobile unit. Nevertheless, little research concerning this issue has been done.

In this thesis, the concept of unit mobility is proposed and viewed as an integration of multiple mobile factors. If the unit mobility of a computer is high, it means that the computer results in a more unstable computing environment for others and, on the other hand, a computer can observe its computing environment by receiving the mobility information from the other computers. A strongly-connected condition plus less movement, less disconnection, and sufficient power may be considered as low mobility. Conversely, if a mobile unit has a low bandwidth, frequent location changes, constant disconnection, and a low power level, it would be considered highly mobile.

The unit mobility is intended to offer more information. With more accurate information, a better decision can be made for better utilisation of system resources. For instance, in the data management area, when the mobility of a mobile unit is low, strict data consistency criteria can be applied because modification on a data item can be immediately propagated to other copies. On the other hand, when unit mobility remains high, a system can switch to a weak data consistency criteria (i.e. inconsistency is allowed during a specified time interval) so as to improve the performance of concurrent access.

It would be ideal if the unit mobility can be quantified, because then the complexity of mobile environments can be better compared. The following sections of this chapter will present the primary results in this research towards this goal.

2.3 Environment Feature Classifications

It should be noted that the term *mobility* in this thesis implies a broad meaning rather than just physical movements of a computer. The unit mobility is described by the behaviour of a mobile unit and its available system resources.

Mobile behaviour of a computer is controlled by its user, called *mobile user*. Two types of factors can affect the behaviour of a mobile unit: *intangible factors* and *tangible factors*. The former one includes working style and psychological role of a mobile user. These types of factors will not be considered in this research. The tangible factors, such as location change, disconnection and moving velocity, are measurable and taken into account in modelling mobility.

Unlike the mobile behaviour, system resources are less controlled by a mobile user. There are many types of system resources, like residual battery life, storage volume, communication bandwidth and so on. The change in either mobile behaviour or system resources causes the changes in the environment of a mobile unit. Therefore, mobile behaviour and available system resources together form a particular environment of a mobile unit.

The salient features that describe such environments, called *environment features*,

must be identified in order to quantify the unit mobility. At present, it is not clear what types of features are essential to support the design of adaptive applications. Different features are chosen for different purposes in existing research. For example, a communication feature (i.e., bandwidth) is identified as a major feature to support multimedia applications [JM96, DBCF94]. In [NPS95] the authors are interested in device features for providing a general system support of mobility. The query optimisation approach in [GA93] relies on residual battery life and workload of system server as major system resource features.

The significance of certain features is dependent upon the objectives and nature of a particular application. The design space of features is potentially very large if possible applications are taken into consideration. As the authors in [NPS95] point out, to utilise mobility information, the mechanism describing mobile environment features should be simple and extensible. Rather than serving a specific application, developing a general scheme for identifying and representing relevant features would be useful. Aiming at this goal, the first step is to classify potential features under some criteria.

Environment features can be classified in different ways. The authors in [NPS95] separate possible features into two categories, called *generic* and *type-specific*, though the author has no explicit intention of feature classification. A generic feature has system-wide meaning and can impact all the users in a system; whereas a type-specific feature is associated with a particular user and has limited effect on applications. For example, the bandwidth is considered as a generic feature because it is a computing resource for all applications. A file reference number is however a specific feature and accessed by relevant applications. It is no longer available once it reaches to its maximum value. Some of the instances of these features and their classifications are shown in Figure 2.2.

It can be seen that this classification is based on the visibility of features. The environment features can also be grouped under the system components because a system can be viewed from different dimensions. Possible system components include

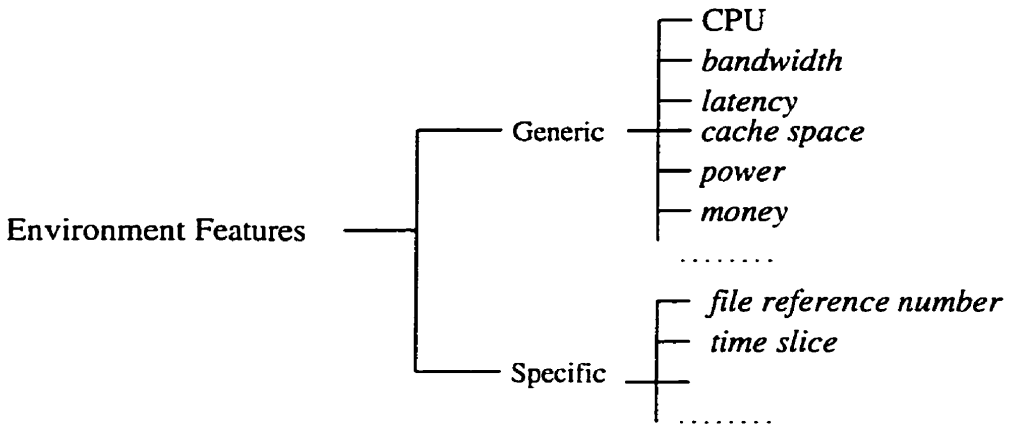


Figure 2.2: Feature classification based on visibility

hardware device component, network communication component, information organisation component, and so on. The environment features are identified and associated with relevant components. For instance, CPU is a device-relevant feature; whereas the bandwidth is network-relevant feature. Figure 2.3 shows a possible classification structure based on this criterion.

Third way of classifying environment features is based on the stability of a feature. If a feature is time independent, it is stable (or static); otherwise it is unstable (or dynamic). For example, the storage volume of a mobile unit is relatively stable and will not change with time. Thus, it is a static feature. On the other hand, the bandwidth, hand-off frequency, and disconnection frequency are examples of dynamic features. From this perspective, the structure of this classification is shown in Figure 2.4.

Among the three classifications, the visibility-based one is bound up with applications. That is, a feature is viewed from the perspective of an application. In fact, type-specific features are also application-specific features. Regardless of these features, all the environment features are in one category. Therefore, the classification does not reveal the information about the difference between features. With the system element-based classification, the difference between features is highlighted. However, it does not identify the common property of features. The stability-based classification is more suitable for design adaptive applications. It separates those

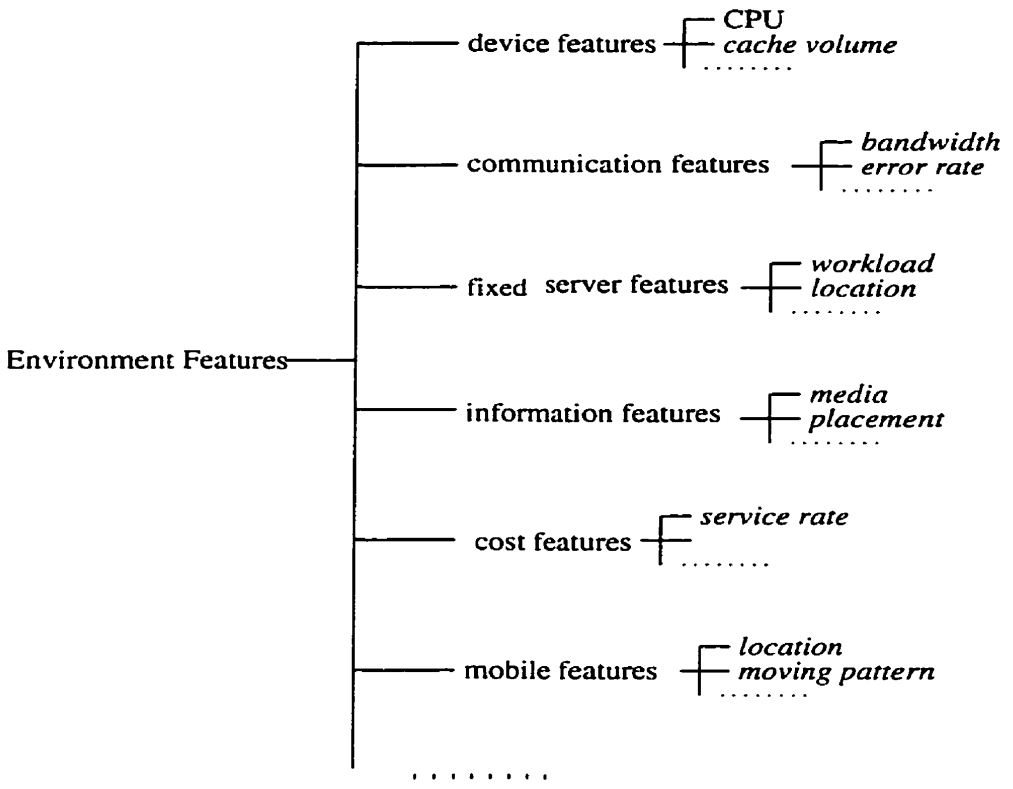


Figure 2.3: Feature classification based on system elements

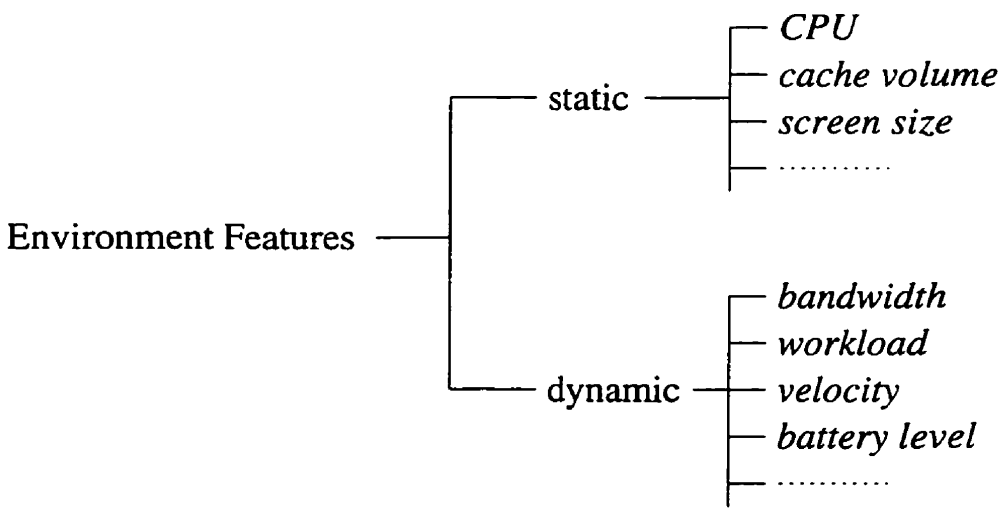


Figure 2.4: Feature classification based on stability

static features from dynamic ones. In a mobile environment, it is dynamic features that provide run-time information. Therefore, dynamic features are more significant. The attention in this work is focused on such features, which are referred to as *mobility features.*

2.4 The Concept of Unit Mobility

In this section, the definitions with respect to unit mobility are presented. The first one identifies features that contribute to mobility.

> **Definition 2.1.** Mobility Feature: An environment feature is defined as a *mobility feature* if the following conditions hold:
>
> **(1)** its state is measurable;
>
> **(2)** its state value may vary continuously with time; and
>
> **(3)** its state change affects computing conditions of other computers.

Residual battery life, disconnection frequency, and location change frequency of a mobile unit are all examples of mobility features. In this research, it is assumed that there exist ways for a mobile unit to be able to capture the information of a mobility feature. Practically, it may require different techniques for different features. For example, for the bandwidth feature, monitoring the data flow of communication ports on a network adapter can be a way to achieve this information [PO95]. Certain disconnection events can be detected by changes in the signal strength, by predicting battery lifetime, or by utilising knowledge of the bandwidth distribution [PB94]. The disconnection frequency may be tracked in software that implements a disconnection counter. To capture the residual battery life, one possible way is to estimate the voltage levels of the battery. The following two definitions build the concept of quantifying the mobility of a computer system (mobile unit):

> **Definition 2.2.** Mobile Status: Let $C =< C_1, C_2, \ldots, C_k >$ be a finite set of mobility features with respect to a computer system S. The value

vector $M_S^t =< c_1, c_2, \ldots, c_k >$ where $c_i \in C_i (1 \leq i \leq k)$ is defined as the *mobile status* of S at time t. We say that $C^K = C_1 \times C_2 \times \ldots \times C_k$ is the domain of M_S^t.

Definition 2.3. Unit Mobility: Let M_S^t be a mobile status of a particular mobile unit S. Then $\Delta = F(M_S^t)$ is defined as the *unit mobility* (UM) of S, where $F()$ is a mapping from C^K to $[0, 1]$, i.e. $F : C^K \rightarrow [0, 1]$. The inverse function of $F()$ exists and is denoted by $F'()$.

The unit mobility can be viewed as an inherent attribute of a mobile unit. The value of this attribute is visible to other mobile units in a system. It is assumed that each mobile unit in the system is constantly publishing its own mobility information. This will allow a particular unit to find out its environment complexity so that a proper action can be taken.

2.5 Representations of Unit Mobility

Suppose all required mobility features are known through measurement or prediction methods. The unit mobility can be expressed in two ways: a *vector representation* or a *single-valued representation.* This section compares the merits and drawbacks of these two representations, and discusses the requirements for a good representation.

A vector representation of unit mobility organises available features into a number of <*feature*,*value*> pairs. The method used in [NPS95] can be viewed as an example of vector representation. With this representation, each component in the vector is independent of others. All the individual features together reflect the status of a mobile unit. In this way, the organisation of information is relatively simple and easy to extend. More importantly, because each individual feature is available, it is able to make use of specific information captured by a particular feature. Therefore, if the design of an application is only interested in the bandwidth feature, it can focus just on this feature.

However, the vector representation is not well suited for the implementation of communication protocols because of variable sized data structures. From the viewpoint of information transmission, mobility information is generally captured from a lower layer of the network protocol stack and passed up to the application layer to support adaptive applications. Because existing communication protocols enforce a limited space for control information, over-sized mobility information may not be acceptable. Even if possible, expanding the size of the control information can result in extra overhead. Taking basic Mobile IP protocol as an example [JM96], 8 or 12 bytes are added to each existing packet to form a new IP header. A shorter header is generally preferred for packet transmission efficiency.

A single-valued representation takes the values of all the mobility features as input and produces a numeric data value as output that denotes a general mobile status. An example of this representation can be found in [PO95]. The following is one of the suggested functions:

$$mobility = k_1 \cdot [1 - \frac{2}{\pi} atan(\frac{\sqrt[4]{\beta}}{10})] \cdot [\frac{1}{\pi} atan(\frac{\varepsilon - 50}{10}) + k_2]$$

where the k_1 and k_2 are coefficient. The β and ε are two measurable mobile factors.

As we can see, given the parameters, the function produces a single value. This value is used to represent the mobile environment at the time when the parameter values are captured. Because a single value can be expressed in a fixed size, it would be easier to incorporate mobility information into an existing protocol.

The implementation of single-valued representation requires a function or mapping that takes a number of mobility features as input and produces a single value as output. Generally, two types of mapping methods can be formed: *non-reversible mapping* or *reversible mapping*.

The mappings proposed in [PO95] are non-reversible ones. They are constructed to maintain proportionality of mobility measurement, i.e., the greater the value, the greater the mobility. Therefore, such mappings can be considered as semantics-preserving. However, in general, for a mapping to hold this property, it must rely on an intricate analysis of the involved mobile features. It would be difficult to design

such a mapping without in-depth understanding of the correlation among individual features, especially when the number of mobility features increases.

With a non-reversible mapping method, the mapping can provide a general image of the current mobile status and make the mobile status comparable. However, this method is more strict in the sense that its output must have a meaningful interpretation that is consistent with commonly used mobility concepts. For instance, considering bandwidth and disconnection frequency, if the bandwidth remains unchanged, the mobility of a mobile unit with less disconnection should be defined as smaller than the one with more disconnection because only such an interpretation is meaningful.

This mapping method is difficult to extend once the model is created. The biggest drawback with this method is that it can only provide an overall image of a mobile status. Because of its non-reversible characteristic, information about individual features is hidden within the model. This can result in difficulties for a recipient of mobility information to make use of specific information about a particular feature. In some circumstances one feature may be more significant than another, such as bandwidth or location change frequency. For instance, if the performance of an application is highly dependent on the quality of communication, it may be only interested in the network bandwidth, instead of the overall mobility.

To overcome the drawback of the semantics-preserving method, a reversible mapping method attempts to make information of individual features available. Therefore, this method involves an one-to-one mapping that simply transforms a mobile status in a multi-dimensional space into one-dimensional scalar space and vice versa. This mapping method leaves the use and interpretation of a mobile status to the recipient of information. Thus, in general, it does not stress the preservation of proportionality of mobility, although it also generates a single-valued output.

In summary, vector representation and single-valued representation of mobile status both have merits and shortcomings. As a good representation method, it should be general enough to include more possible mobility features. It should also take

limited space and be able to incorporate into existing protocols. More importantly, it should be flexible enough to provide specific mobility information that an application requires. In short, to exploit the advantages from each method, a combined scheme is required. This scheme should hold the following properties:

(1) efficiency for transmission;

(2) extensibility in structure; and

(3) visibility of individual features.

Generally, a single-valued representation with a reversible mapping technique can meet these requirements. In the next section, such a mapping scheme is presented.

2.6 The Mapping of Unit Mobility

The goal of constructing a mapping function for unit mobility is two fold. First, it must combine individual mobility information for efficient transmission, and second, it must determine information contributed by each mobility feature when combined information is received. This means that a reversible mapping is needed.

The mapping method is based on the conversion of numeric systems. Let k be the number of mobility features. First, the numeric domain of each mobility feature is separated into r sub-ranges according to the application requirements. The variable r is referred to as an *accuracy factor* of mobile status measurement. Since a given value of each mobility feature must fall in one of the sub-ranges, then a scaled vector $< v_0, v_1, \ldots, v_{k-1} >$ can be obtained, where v_i stands for the corresponding number of sub-range of the i_{th} mobility feature.

The variable r is used as the base of a numeric system so that a one-to-one mapping from an r-based system to a 10-based system can be implemented. The unit mobility, thus, can be computed with the following formula:

$$\Delta = F(v_0, v_1, \ldots, v_{k-1}) = \frac{\sum_{i=0}^{k-1} v_i r^{k-1-i}}{\sum_{i=0}^{k-1} (r-1) r^i}$$

To illustrate the mapping method and process, an example is given below to explain both the concept of unit mobility and the mapping process.

Suppose that captured mobility information comes from a few measurable mobility features: bandwidth(B), disconnection ratio(D) and residual battery life(R). Thus, the vector $< B, D, R >$ describes the mobile status of a mobile unit.

Network bandwidth is measured in bits per second (bps). Greater bandwidth indicates faster data-transfer capability. A disconnection ratio describes the behaviour of a mobile unit with respect to its disconnection history. In this work, it is defined as an average proportion of disconnection duration in a specified time interval. Let m be the total number of disconnection in a time interval $t_1 - t_0$ and d_i the disconnection duration at the ith disconnection. Then $D = \frac{\sum_{i=1}^{m} d_i}{t_1 - t_0}$. The residual battery life is estimated with the voltage levels of the battery.

Let the domains of bandwidth, disconnection ratio, and residual battery life be $0 \sim 1000Kbps$, $0 \sim 1$, $0 \sim 12$, respectively. To do the mapping, the domains are first divided into small segments. Here let's choose an accuracy control factor r to be 100 provided that this makes a sub-range small enough to distinguish two different status values. Therefore, each domain has 100 sub-ranges, which are labelled with sequential numbers (see Figure 2.5).

Note that the order of the sequential numbers of sub-ranges for each mobility feature in Figure 2.5. To preserve the semantics of mobility, a greater number should denote a greater mobility. Taking bandwidth as an example, a higher bandwidth generally means stronger network connectivity and hence less mobility. Conversely, a lower bandwidth means higher mobility. Therefore, in the example, the order of numbers of sub-ranges is changed to reflect this fact.

Suppose at a particular time point, a mobile unit detects that the current bandwidth is 64Kbps, disconnection ratio is 0.3, and residual battery life is 10. Then a numeric vector <64,0.3,10> reflects the mobile status of the mobile unit at this specific point.

Looking at Figure 2.5, it can be seen that 64Kbps falls into 93^{rd} sub-range in

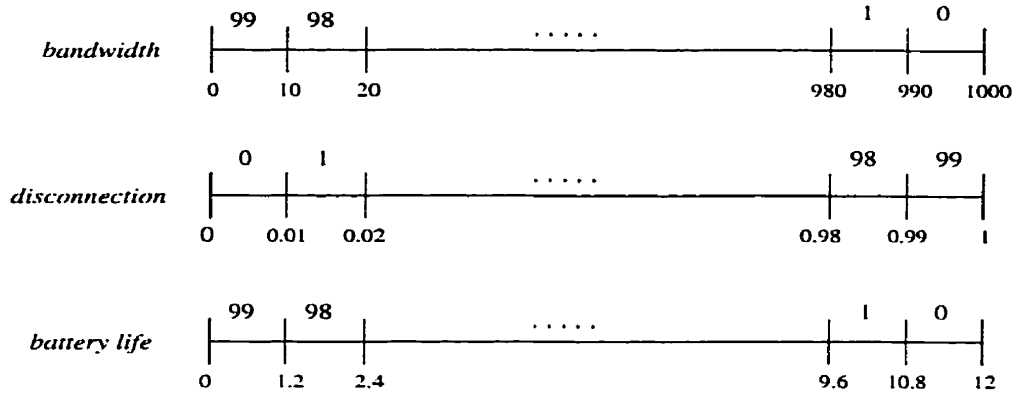


Figure 2.5: The example of context feature mapping

bandwidth dimension. Similarly, 0.3 falls into 29^{th} sub-range of disconnection ratio dimension and 10 falls into the 1^{st} sub-range of battery life dimension, respectively. Thus, we have a scaled vector, or *scaled mobile status* as follows:

$< v_0, v_1, v_2 >=< 93, 29, 1 >$,

Using the mapping function, we have a combined representation of mobile status:

$$\Delta = f(93, 29, 1) = \frac{(93 \times 100^2 + 29 \times 100^1 + 1 \times 100^0)}{\sum_{i=0}^{2} 99 \times 100^i}$$

$$= \frac{932901}{999999} = 0.9329019 \approx 0.9329$$

As described early, the mapping result generally does not require a meaningful interpretation. However, for a small number of mobility features, it is possible to preserve proportionality of mobility with a deliberate design. A deliberate design means a proper-scaled vector of mobile status plus the proper order of vector elements. For instance, in the above example, the order of mobility features in a mobile status vector is deliberately arranged. The order acts as a weight for the significance of each feature. In the example, the order of significance of the elements in the vector is from left to right in the formula (top to bottom in Figure 2.5). In other words, bandwidth is viewed as the most significant factor, while residual battery life as the least one. However, other applications could define a different order based on the significance of the parameters.

If a single value produced from the mapping maintains the proportionality of

mobility, it describes the general mobility of a mobile unit. If a user is not interested in individual features, then this value still gives some useful information. For instance, if the bandwidth goes up, then the general mobility drops accordingly.

With the mapping method above, the accuracy factor r is a system-wide parameter and must be pre-determined. Since the definition of $F()$ is a one-to-one mapping based on numeric system conversion and its reversible mapping $F'()$ exists, with knowledge of r, mobility information derived from each individual feature can be restored.

One implication with this method is that if both r and k are large, it can result in a large intermediate computing result. Nevertheless, in practice the number of mobility features supporting adaptive applications is expected to be small. If this is not true, a multi-level mapping may be applied.

2.7 Unit Relative Mobility

The information derived from unit mobility can be used for different purposes. An application can react to its environment changes directly based on available mobility information. It can also be used to support more complicated mobile applications. For instance, in an *ad hoc* mobile network environment, all the computers can be mobile. Therefore, a communication route must be dynamically established. One of the techniques suggested in [ABB+96] is to partition the network into several clusters. In each cluster, a unit with the lowest ID is elected as a cluster head based on periodically published ID information. Alternatively, if mobility information is available, it can be used to cluster the mobile units where the unit with the lowest mobility may be selected as the router, because it is more likely to stay active. In this section, the measure of relative mobility is addressed.

Suppose unit mobility can be published by a mobile unit and the other units can receive such information. Then a unit can use the information to determine its mobility relation to others. In general, let $U = \{u_1, u_2, \ldots, u_n\}$ be n mobile units in a mobile distributed system. The event that u_i receives the unit mobility published by

u_j is denoted by $< u_i, u_j >$. In this case, u_i is called an *observer*, and u_j a *publisher*. Each of them maintains its own unit mobility, or Δ value. An observer u_i combines its Δ with its publisher's to come up with a relative mobility measure between them.

Definition 2.4. Unit Relative Mobility: Let u_i be an observer and u_j its publisher, where Δ_i and Δ_j are their unit mobilities, respectively. $\delta_{ij} = G(F'(\Delta_i), F'(\Delta_j))$ $(\delta_{ij} \leq 1)$ is defined as the *unit relative mobility* (URM) between u_i and u_j, where $G()$ is given by

$$\begin{aligned}\delta_{ij} &= G(F'(\Delta_i), F'(\Delta_j)) \\ &= G(< v_{i0}, v_{i1}, \ldots, v_{ik} >, < v_{j0}, v_{j1}, \ldots, v_{jk} >) \\ &= \frac{\sum_{t=0}^{k-1} abs(v_{it} - v_{jt}) r^{k-1-t}}{\sum_{t=0}^{k-1} (r-1) r^t}\end{aligned}$$

Given $F'(\Delta_i) =< v_{i0}, v_{i1}, \ldots, v_{ik} >$ and $F'(\Delta_j) =< v_{j0}, v_{j1}, \ldots, v_{jk} >$, we can obtain the unit relative mobility. For example, suppose $F'(\Delta_i) =< 89, 1, 13 >$ and $F'(\Delta_j) =< 80, 2, 10 >$, and assume $r = 100$, then:

$$\begin{aligned}\delta_{ij} &= G(< 89, 1, 13 >, < 80, 2, 10 >) \\ &= \frac{(89-80)100^2 + (2-1)100^1 + (13-10)100^0}{999999} \\ &= 90103/999999 = 0.090103 \approx 0.0901\end{aligned}$$

Theorem 2.1. Unit relative mobility holds the properties of reflexivity, symmetry, and transitivity. That is,

(1) $\delta_{ii} = 0$,

(2) $\delta_{ij} = \delta_{ji}$, and

(3) if $\delta_{ij} \geq \delta_{ju}$ and $\delta_{ju} \geq \delta_{uv}$, then $\delta_{ij} \geq \delta_{uv}$.

Proof: The correctness of the first two properties can be seen directly from the definition of unit relative mobility. The third property is straightforward because δ_{ij}, δ_{ju}, and δ_{uv} are numeric data items between 0 and 1.

If $\delta_{ij} = 0$, it means that the mobility status of two mobile units are very close (the closeness is dependent on the selection of the accuracy control factor r). Following the previous example, an intuitive interpretation for given mobility features is

that the observer and its publisher have similar communication bandwidth, similar disconnection ratio, and similar residual battery life.

However, it should be noted that equal unit relative mobility does not mean that the unit mobility of mobile units are identical. For example, suppose that $< 50, 3, 10 >$ and $< 40, 1, 5 >$ are the scaled mobile status vectors for two mobile units at time t_0. At time t_1, their scaled mobile status vectors change to $< 30, 4, 11 >$ and $< 20, 2, 6 >$, respectively. By Definition 2.4, the unit relative mobility of the two mobile units are identical. However, it can be seen that at time t_1 the environment of the units are less mobile than at t_0.

In fact, unit mobility can be viewed as a special case of unit relative mobility. That is, an observer always takes a fixed computer as its publisher. A fixed computer is assumed to have a unit mobility of zero. From this perspective, unit mobility measures the mobile status of a mobile unit relative to a fixed system.

The concept of unit relative mobility can be further extended to measure the mobile status of a unit relative to a group of mobile units. In general, a $n \times n$ matrix for a system with n mobile units can be generated that describes the mobile relationship between any two units.

	u_1	u_2	u_3	...	u_n
u_1	0	δ_{12}	δ_{13}	...	δ_{1n}
u_2	δ_{21}	0	δ_{23}	...	δ_{2n}
u_3	δ_{31}	δ_{32}	0	...	δ_{3n}
	...	...	...		...
u_n	δ_{n1}	δ_{n2}	δ_{n3}	...	0

Each row in the matrix is a notion of mobile environment perceived from a particular mobile unit (observer). Therefore, given a vector $< \delta_{i1}, \delta_{i2}, \ldots, \delta_{in} >$, an integrated measure, called *environment mobility*, can be defined for an observer u_i.

Definition 2.5. Environment Mobility: Suppose that $U = \{u_1, u_2, \ldots, u_n\}$ are n mobile units in a system. Accordingly, the unit relative mobility

to any unit u_i is given by $\delta_i = \{\delta_{i1}, \delta_{i2}, \ldots, \delta_{in}\}$. Environment mobility perceived by u_i, denoted by $\overline{\delta_i}$, is defined as

$$\overline{\delta_i} = \frac{\sum_{j=1, i \neq j}^{n} \delta_{i,j}}{n-1}$$

As an example, suppose u_1 is an observer, and

$$\delta_1 =< \delta_{12}, \delta_{13}, \delta_{14}, \delta_{15}, \delta_{16}, \delta_{17} >=< 0.2506, 0.7045, 0.0000, 0.1054, 0.6087, 0.4539 > .$$

Then,

$$\overline{\delta_1} = \frac{0.2506 + 0.7045 + 0.0000 + 0.1054, +0.6087 + 0.4539}{6} = 0.35385$$

Group relative mobility measures average mobility of a (sub)system from the viewpoint of a particular mobile unit. The definition of group relative mobility is consistent to the unit relative mobility. It can be seen that if all the publishers in a system are fixed, the group relative mobility of an observer becomes its unit mobility because the unit relative mobility between the observer and any publisher is equal to the unit mobility of that observer.

As mentioned earlier, mobility information may be used for different application purposes. In this research, we focus its application on database management. The issue of optimising mobile transaction processing with unit mobility information is studied in the following chapters.

Chapter 3

AN ADAPTIVE TRANSACTION SCHEDULING APPROACH

Mobile transaction processing is more complicated than transaction processing in a traditional computing environment because it must take into account the additional characteristics of a computing environment with low bandwidth, unreliable communication, frequent disconnection, varying locations, limited services, and heterogeneous networks. At present, no research has been done to use mobility information to improve the performance of transaction management. This chapter presents the primary research results in this direction.

3.1 Problem Analysis

3.1.1 Existing Transaction Processing Models

To illustrate the model presented in this thesis, this section begins by examining transaction processing in a traditional distributed database system.

Informally, a *transaction* can be defined as a sequence of *read/write* operations on one or more data items in a database system that potentially change the state of the database system and transform the database from one consistent state to

another. In a traditional distributed database environment a transaction is a basic unit of consistent and reliable computing and characterised by *atomicity*, *consistency*, *isolation*, and *durability*, i.e. ACID properties [OV91].

The ACID properties ensure that, firstly, a transaction either carries out all of its operations or none at all. When an error occurs, a transaction is a basic unit of error recovery. Secondly, the completion of each transaction (*commit* or *abort*) must leave the database in a consistent state. That is, all the replicated data keeps identical, and meets pre-defined consistency constraints. Thirdly, although it is possible that during the execution of a transaction there exists temporary data inconsistency, this temporary inconsistency should not be visible to other transactions. Finally, once a transaction is committed, its effect must be permanently recorded and survive any failures.

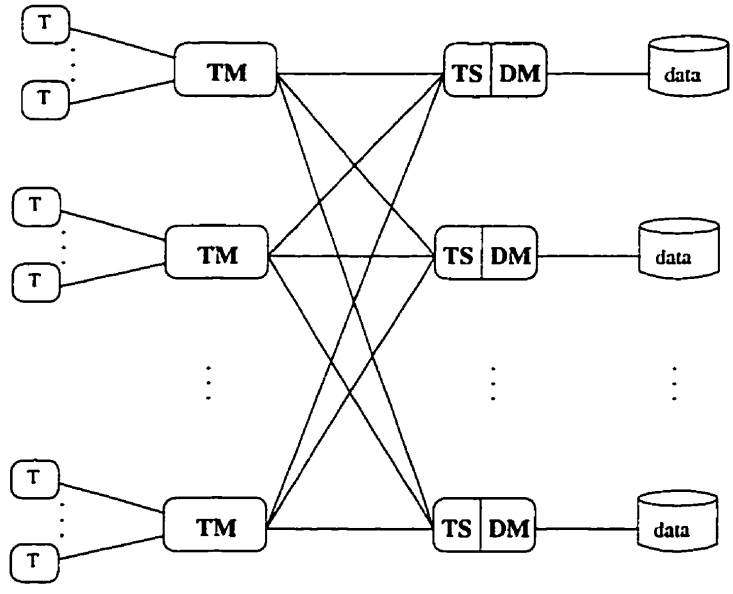


Figure 3.1: Major software components of distributed transaction processing

The goal of transaction processing is to keep the database in a consistent state even when concurrent access and failures occur. The execution of a transaction in a distributed environment is a co-operative process among software components: namely, transaction manager(TM), transaction scheduler(TS), and data manager(DM).

A TM is responsible for co-ordinating the execution of the transaction operations

on behalf of an application. As a co-ordinator, it receives transaction operation requests from a user process and monitors the operating status performed by the other components that are running on the other computers (i.e. participants). A TS takes the responsibility for the implementation of a specific concurrency control algorithm for synchronising access to the database. A DM runs on a host where shared data is stored and performs actual read/write operations on its portion of the database. The relationship among these components is illustrated in Figure 3.1.

If there is no replication of data, the standard process of distributed transaction processing works as follows [CL91]: upon receiving a transaction request the TM acting as a co-ordinator directs the operations to involved participants, which own the TS and DM components on the other sites. When a participant finishes executing its operations on a local workspace, it sends an execution complete message to the co-ordinator. When the co-ordinator has received this message from all the participants, it starts a *two-phase commit (2PC)* protocol by sending prepare messages to all participants. If a participant is ready to commit, it sends a prepared message back to the co-ordinator. After receiving prepared messages from all participants, the co-ordinator sends a commit message to each participant and thus initiates the second phase of the protocol. Upon receipt of this message, each participant makes its operations on the local workspace permanent and sends a committed message back to the co-ordinator. The transaction is successfully committed after the co-ordinator has received the committed message from all the participants. If any participants cannot commit, it will send an abort message to the co-ordinator in the first phase of the protocol. This makes the co-ordinator in the second phase send an abort message instead of commit message to all the participants. Hence, the entire transaction is aborted.

In a distributed environment, multiple users access a database simultaneously. To prevent interference among users and enhance the performance of transaction processing, concurrent accesses must be synchronised to ensure database consistency. Concurrency control algorithms follow a specified correctness criterion. The most

widely used correctness criterion is conflict serializability. It says that the interleaving execution of two transactions should be equivalent to their sequential execution in some order. According to the time point of synchronisation in the process of transaction execution, concurrency control algorithms can be distinguished into pessimistic and optimistic ones.

A pessimistic algorithm ensures there are no conflicting operations before using the data and performs data consistency validation first. By conflicting operations it means that two operations act on the same data item and one of them is a write operation. Typical pessimistic algorithms are based on *two-phase locking (2PL)* synchronisation technique [Sto94]. With the 2PL technique, before reading a data item, a transaction must "own" a readlock on that item and before writing into a data item, it must "own" a writelock on that item. Whether a readlock or writelock can be granted is governed by specified rules. If there are multiple copies of a data item, to read the data it suffices to set a read lock on any copy of the item; to update an item, write locks are required on all copies. If the requested lock cannot be granted, the transaction is blocked on a write request and the operation is placed on a queue waiting for all the copies of the desired data item to be unlocked.

It can be seen that pessimistic synchronisation techniques aim at maintaining strict data consistency, but they can reduce concurrency if a lock cannot be released quickly. In contrast, an optimistic algorithm for concurrency control uses the data first without co-ordination and delays the validation until the update of data occurs. At this point, if a conflict is discovered, previous operations must be rolled back and the transaction is aborted and possibly restarted.

Optimistic two-phase locking (O2PL) is an example of an optimistic algorithm [CL91]. This algorithm is designed to reduce the message communication between a co-ordinator and involved participants when the replication of data items exist. Without replication, O2PL and 2PL are in fact identical. However, if replication is present, a write lock in O2PL is immediately granted on the local copy of the item, but the lock requests on remote copies are deferred until the beginning of the commit

phase. In contrast, 2PL must ask for the write locks on all the copies of a data item. Thus, communication with remote computers is reduced. When a transaction reaches a commit point, the write lock requests on remote copies are combined into the message of the commit protocol. At this time point, the consistency of replica must be verified. If all the data items are valid, the transaction can commit; otherwise, the transaction is aborted.

3.1.2 Limitation of Conventional Transaction Model

One limitation of the existing transaction model is that it lacks the satisfactory ability to handle weak connections and support autonomous operations when the communication of a mobile unit on the wireless network is unstable. *Autonomous operation* is a mode of operation in which a mobile unit continues to use available (maybe stale) data to perform transaction processing during the time interval of a disconnection. Supporting autonomous operation is desired in a mobile environment [PB95, WC95]. The benefits supporting autonomous operations are recognised in [PB94, PB95], and relate to techniques such as extending battery life by avoiding wireless transmission, reducing network charges, and maintaining radio silence for security reasons. Since weak disconnection or total disconnection are common occurrences in a mobile environment it is not appropriate to simply treat these events as abnormal conditions like they are in the conventional distributed environment.

An example below is used to illustrate the limitation of existing transaction models. It is assumed that a distributed database system is partitioned into four nodes in a network: A,B,C and D, shown as Figure 3.2. Node A is further assumed to be the transaction starting node.

To manage distributed transactions, a co-ordinator is required. In our example, the co-ordinator can be designated as either A or B. In a traditional (non-mobile) environment, the selection of A or B as a co-ordinator does not make much difference in terms of data consistency control. Since the network between A and the rest of the network is always connected and the bandwidth between two nodes is relatively high,

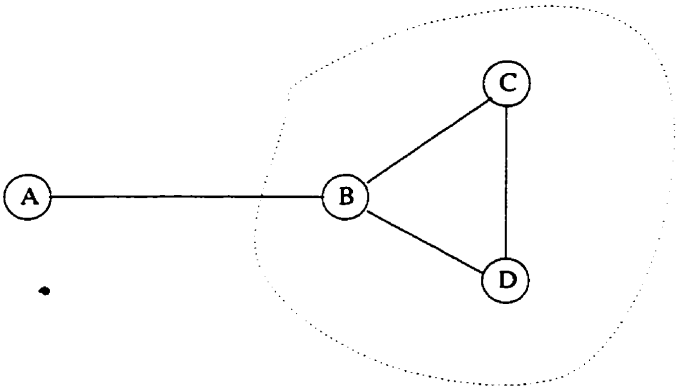


Figure 3.2: An example of traditional distributed database systems

the messages that control transaction execution can be sent back and forth between a co-ordinator and a participant within an expected time interval. If the connection is broken, the underlying assumption is violated and a communication failure occurs.

However, this scenario is no longer valid in a mobile environment. Now suppose that A is a mobile unit and that the communication between A and B is through a wireless network. The bandwidth between them is likely varying. In the worst case, it can be zero, i.e., disconnected. In such an environment, no matter which node is selected to be a co-ordinator, no optimal system performance nor can data consistency be guaranteed.

Let us further look into this case. Suppose that the fixed node B is designated to be a co-ordinator, and that A,C and D are participants for a given transaction. Assuming that initially A is connected to the network, a mobile transaction is started by A and submitted to a co-ordinator, i.e., node B. Then B has to contact each node, including A, where required data is stored. To manage concurrency and maintain strict consistency, the data items accessed by the transaction must be locked. At this time, if the bandwidth between A and B is very low, the communication will suffer from a long delay. If the network is disconnected, the locks cannot be released so that those data items are not available for other transactions. This does not only affect the response time of current transactions, but also the throughput of the entire system.

Now let us consider if the mobile unit A is designated to be a co-ordinator. When a mobile transaction is initiated by A, as a co-ordinator, it has to contact all the participants. Suppose, at this time, A is connected. If the bandwidth will remain high, then there would be no serious problem applying the traditional processing method. However, if the bandwidth deteriorates after the transaction is started, the same problems as the scheme with a fixed co-ordinator B occurs.

In short, under the existence of network disconnection, no matter which computer serves as a co-ordinator, once the connection is broken, the computation process with traditional schemes can no longer carry on.

3.1.3 Requirements of Mobile Transaction Processing

Varying bandwidth, short battery life, and frequent relocation of mobile units are three novel factors introduced by the mobile environment. In such an environment, the task of achieving the required performance and ensuring the consistency of shared data becomes more difficult than in a traditional environment.

Mobile database applications usually take place in the context of a larger enterprise. Mobile users will desire access to private (local) or enterprise (global) databases. However, they often deliberately avoid the use of the network resources to reduce cost or power consumption, or simply because no networking capability is available at their current location. As mentioned in Chapter 1, there are two types of disconnection: *predictable* and *non-predictable*. A predictable disconnection is planned by a user ahead of time and thus an explicit disconnection protocol can be conducted before the disconnection takes place. On the other hand, a non-predictable disconnection may be out of the awareness of a user and cannot be distinguished from a long-lived transaction [EJB95].

It is possible that the part of a computation executing on a mobile unit may continue executing while the mobile unit is not connected to the network. To support this type of computing requirement, disconnection must be handled elegantly and cannot simply be treated as a failure [WC95, PB93] as it is in the conventional

distributed database environment. To handle disconnection, a general strategy is to increase resource availability by the replication of data at multiple sites. However, replication requires more sophisticated data consistency control methods. Since the effect of local updates during disconnection cannot be reflected immediately on replicated copies, strict data consistency criteria that require all replicas to be identical at any given time are not suitable. Thus, some weak data consistency (i.e., some replicas may be inconsistent at a certain time interval) must be tolerated in a mobile environment.

In summary, a practical scheme for mobile transaction processing should meet the following requirements:

(1) supporting the autonomous operations of mobile units during disconnection [PB94, PB95], and

(2) maintaining data consistency under expected performance criteria.

It is difficult to meet these two requirements because they are fundamentally conflicting. For example, increased autonomy of mobile units requires more independent operations on a mobile unit, but this results in more difficult data consistency control and in turn reduces system performance. Therefore, a balance must be made in the design of a mobile transaction management approach.

An important issue in a mobile environment is that mobile units can be in varying states. A uniform strategy of transaction management is not appropriate to all mobile units. Mobility requires adaptive procedures for individual mobile units. In some mobile computing areas, like network protocols [JM96, DBCF94, Yua93], file systems [MES95], information systems [Kat94, PB94, IV94], and multimedia applications [ABB+96, DBCF94], the design of system procedures has already taken into account specific mobility information, such as network bandwidth, residual battery life, and message arrival frequency to improve overall system performance.

As a major computing area, transaction processing in database management may benefit from captured mobility information as well. The approach presented in the

following sections can be considered as the first step towards this direction.

3.2 Potential Policies of Transaction Execution

The example in Section 3.1.2 shows that weak disconnection or total disconnection makes traditional transaction management schemes unsuitable for mobile transaction processing. The major problem results from the need to maintain interactions between a co-ordinator and the transaction participants during the process of transaction execution.

As pointed out early, if the network is partitioned, a fundamental technique for non-disruption of transaction processing is to make the data available locally. Following the previous example, if a mobile transaction needs both local data (mobile data) and remote data (fixed data), two basic policies can be used to carry out data replication.

The first policy is called *transaction shipping policy (TSP)*, or *uploading policy*. That is, for a given transaction request, the transaction with a copy of required data on the mobile unit is transferred to the fixed host. The fixed host co-ordinates the execution of the transaction on behalf of the mobile unit and returns the final results to the mobile unit. In this case, a mobile transaction can be processed just like a traditional distributed transaction. The only difference is that the propagation of modifications and the return of results to a mobile unit may be delayed until the mobile unit is reconnected at a desired bandwidth.

The transaction shipping policy is relatively simple. Its main advantages are :

(1) *maintaining strict data consistency* because an entire transaction is managed and executed on fixed computers, traditional transaction schemes can be applied.

(2) *reducing battery consumption* because the operations are shifted to the fixed network, computation tasks on a mobile unit are avoided.

(3) *increasing the opportunities of concurrent access* because long delays from poor communication is avoided, more access on a data item is allowed.

However, a shortcoming of transaction shipping policy is that no autonomous operations are allowed during the disconnection of a mobile unit because required data on the other nodes is not available. This policy is suitable when the entire database is allocated on the fixed machines. In other words, a mobile unit only plays the role of a client, or as a remote terminal [Nar94]. In fact, this specific processing model has been taken as a underlying assumption by some researchers [YZ94b, YZ94a].

To allow continuous computation when a disconnection occurs, the data stored at the fixed hosts can be duplicated on a mobile unit rather than moving data to a fixed host. Instead of making a full replication on a local disk, using cached copies is a common technique to support autonomous operations and minimise network access. This approach downloads data from the fixed computers to the mobile unit and is called *data shipping policy (DSP)* or *downloading policy.*

Since data involved in a mobile transaction is always locally available, this policy allows transaction processing independent of fixed data services. Hence, the autonomous operations can be carried out at the mobile unit. In addition, this policy uses less battery power compared to the TSP policy with respect to data transmission because the network connection between a mobile unit and a fixed host is asymmetric. A fixed host typically has a stronger transmitter and unlimited power. Therefore, transporting data from a fixed host to a mobile unit is likely to be more efficient than transporting data in the opposite direction.

However, the data shipping policy implies two issues. The first one is how to download required data. Replication can be created in a planned mode. That is, required data is transferred from fixed hosts to a mobile unit before a disconnection occurs. This mode requires that a disconnection protocol knows what data will be used in the near future. Conversely, replication can also be created in a non-planned mode. That is, data is downloaded based on current transaction requirements and network conditions. For instance, if a transaction uses data items X and Y, and at this time point the bandwidth is high, then the data will be transferred to the

mobile unit. Considering two types of disconnection, the first mode is more suitable for predictable disconnection; while the second one is suitable for non-predictable disconnection.

Another issue is how to maintain data consistency. Replicated data on a mobile unit raises resource availability, but at the price of a more complicated data consistency protocol. This issue will be further addressed in the following sections.

In summary, due to disconnection, a mobile transaction scheme that relies on constant network communication is no longer appropriate. A transaction shipping or data shipping policy becomes a good candidate for mobile transaction processing, although each one has its own advantages and shortcomings.

3.3 Adaptive Transaction Scheduling

3.3.1 Architecture and Assumptions about Mobile Database

As discussed in Chapter 2, there can be different models for mobile computing systems. Among them, one model is most likely to be targeted as a computing environment for mobile database systems. Its position in the design space of mobile computing systems is illustrated in Figure 3.3.

The position means that the entire computing system consists of both wired and wireless networks. In the system, some computers are fixed and some are mobile. Among the mobile computers, some play the role of mobile client and some the role of mobile host.

This architecture is widely accepted in existing research for mobile database applications [EJB95, WC95, IB92, MHB97, MDC93], where a global database is distributed among the fixed network nodes. An example of such an architecture is shown Figure 3.4.

In this system architecture, all the network nodes in the wired part are fixed units. Mobile units retain communication through the wireless links. Some fixed units, called *base stations* or *mobile support stations*, have special functionality with

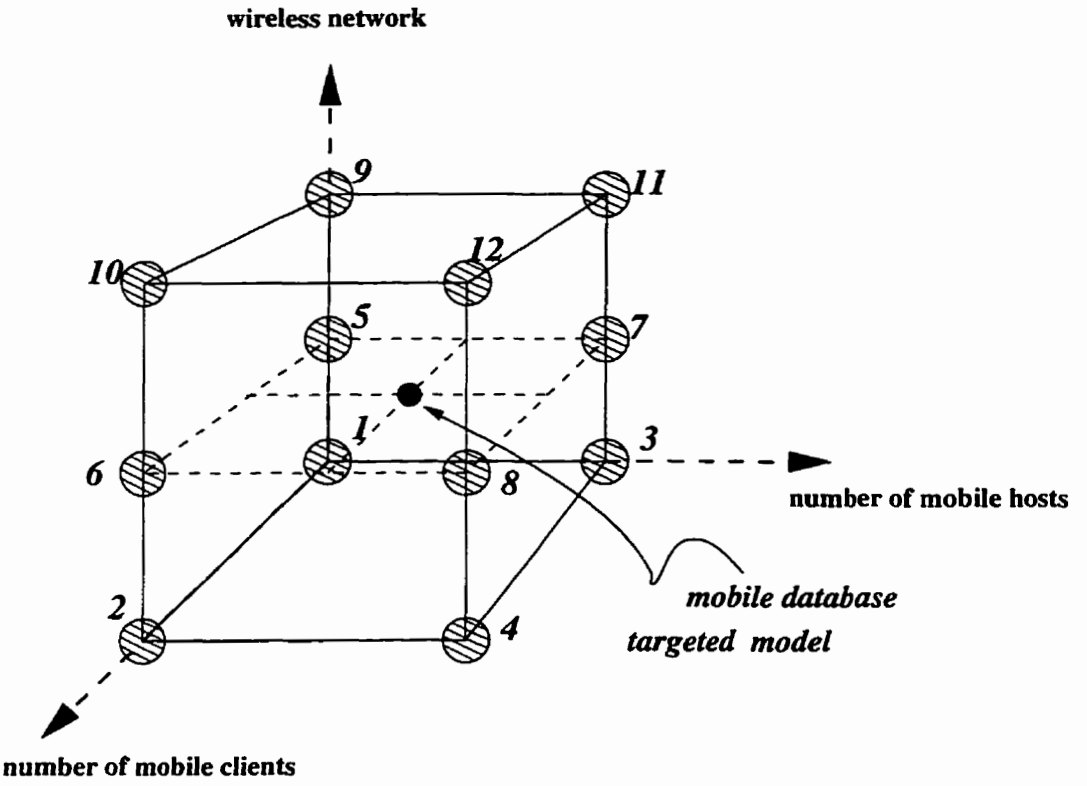


Figure 3.3: A network model targeted by mobile database systems

a wireless interface to communicate with mobile units. Each mobile station provides networking services for all the mobile units within a given geographic area, known as a *cell*. In other words, each cell has a base station and mobile units within that cell access data on remote nodes through the local base station.

From the standpoint of a base station, the entire system can be viewed as an integration of two sub-systems. The first is a sub-system between mobile units and a base station called the *mobile sub-system*. The second is a sub-system between a base station and the fixed hosts, called the *fixed sub-system*. The computers in the fixed sub-system are connected with physical lines all the time; while the computers in the mobile sub-system are often disconnected from the network. Figure 3.5 shows the components of two sub-systems and their relationships.

The two sub-systems perspective of mobile databases is used to describe the *Adaptive Transaction Scheduling* (ATS) approach developed in this thesis. Unlike existing research, it is assumed in this work that a global database is distributed

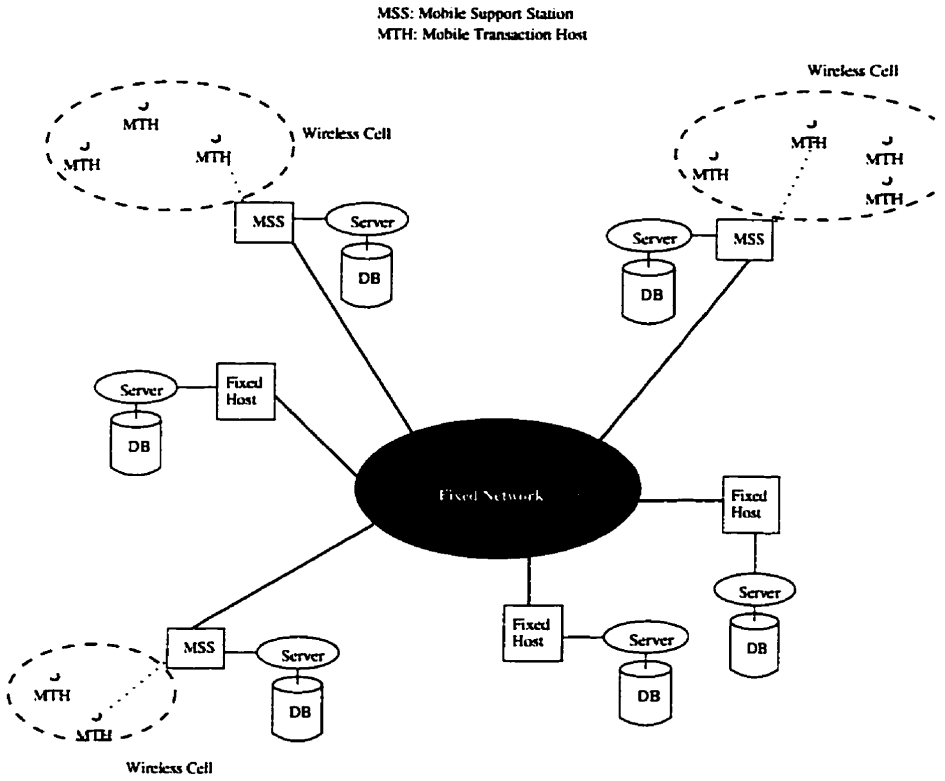


Figure 3.4: Architecture of mobile database systems

both on the fixed network nodes and on some mobile computers that play the role of mobile hosts. A more precise description is given below.

A *global database (GDB)* is defined as a finite set of data items. A GDB is partitioned among fixed hosts as well as mobile hosts. The data items on fixed hosts make up the *fixed database (FDB)* of the GDB and the data items on mobile hosts make up the *mobile database (MDB)* of the GDB. Moreover, the data items on a single mobile host are called a *mobile part* of the MDB. Accordingly, a data item in the FDB is referred to as a *fixed data item* and a data item in the MDB is referred to as a *mobile data item.* Furthermore, each data item has one primary copy and thus one owner. Only its owner can update the primary copy of a data item. A data item in the MDB is owned by a mobile host and a data item in the FDB is owned by a fixed host.

A *mobile transaction host* is a mobile host that initiates a mobile transaction. A *mobile transaction* is informally defined as a set of database operations that access

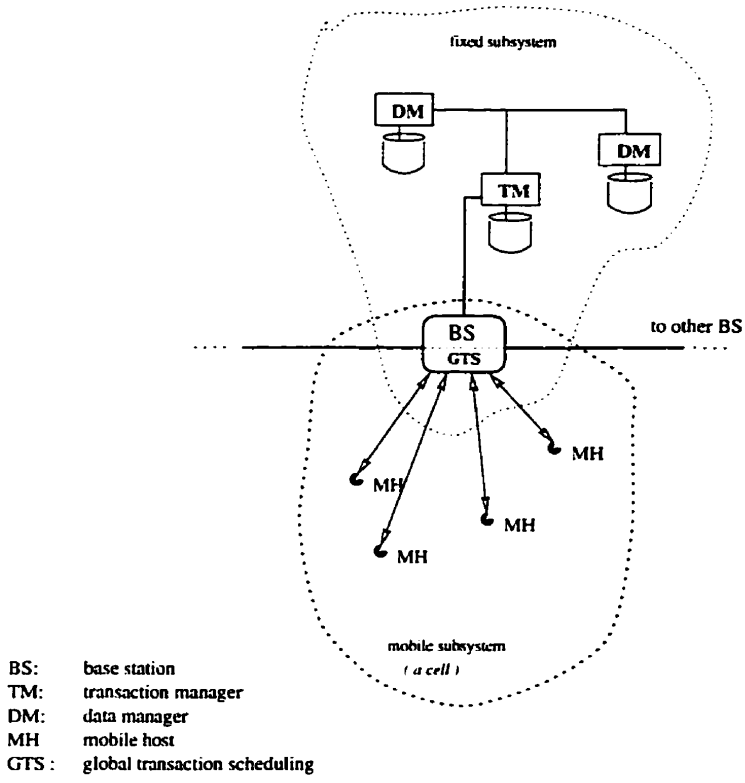


Figure 3.5: The view of two sub-systems of a mobile database

the data items in the FDB and/or the MDB.

In this work, a mobile host is assumed to have local storage and computing capability, and is able to capture and publish mobility information about itself. The fixed data items are replicated by caching on a mobile host, whereas the mobile data items have a full replication on a fixed host. Moreover, a mobile transaction does not access the data on other mobile units. That is, it is restricted to access only local mobile data items and fixed data items.

Concerning the architecture of the mobile database system, each base station is assumed to have an attached database server, named a *Global Transaction Scheduler* (GTS) (see Figure 3.5). It can receive transaction requests, make decisions, transfer data, and communicate with a conventional transaction manager that co-ordinates the execution of a transaction on the fixed sub-system.

3.3.2 Approach Description

In Section 3.2, two policies were identified as candidates for mobile transaction processing that avoid performance penalties due to weak network connection or disconnection. If a mobile transaction is always processed with a data shipping policy, it is known as a *fixed data shipping (FDS)* approach. Conversely, if a mobile transaction is always processed with a transaction shipping policy, it is known as a *fixed transaction shipping (FTS)* approach.

The discussion in Section 3.2 shows that neither FDS nor FTS on its own can produce better performance in terms of system throughput and battery consumption in all situations. Since mobile transaction processing is a process of dynamic decision-making, rather than using a fixed transaction processing approach, an adaptive approach to control the execution of a transaction is introduced. That is, taking the data shipping and transaction shipping policies as two basic options, a decision is made for a given transaction request based on available mobility information to select a transaction processing policy optimal to the current computing environment.

With the adaptive approach, a mobile transaction can be scheduled with a data shipping policy or a transaction shipping policy. If a mobile transaction is scheduled with a data shipping policy, it is called a *data shipping transaction*, or *DS transaction*. Similarly, if a mobile transaction is scheduled with a transaction shipping policy, it is called a *transaction shipping transaction*, or *TS transaction*. The basic concepts of the ATS approach are illustrated in Figure 3.6.

A mobile transaction MT (DS transaction or TS transaction) in the ATS approach is decomposed into two sub-transactions: namely, a *master sub-transaction* T_{master} and a *slave sub-transaction* T_{slave}. Corresponding to two sub-transactions, there are two commit points: *local commit* and *global commit*, respectively. A local commit records the status of a master sub-transaction processing tentatively when the processing of a master sub-transaction is done, but a slave sub-transaction cannot be issued due to network disconnection at this time. A global commit implies that a slave sub-transaction has been executed and modified data has been prop-

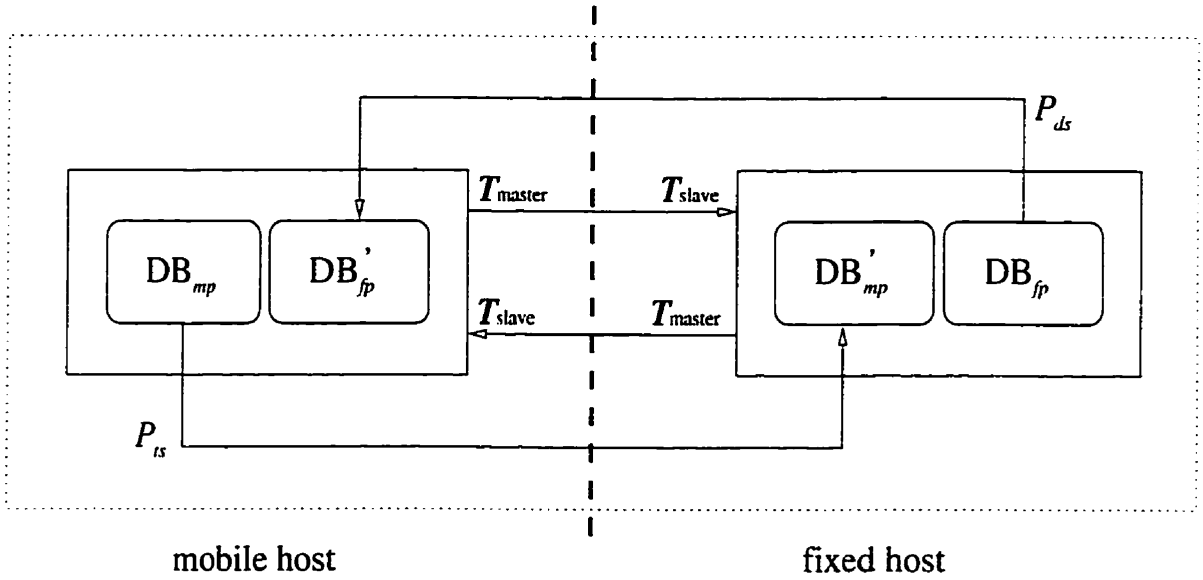


Figure 3.6: Two-phase mobile transaction processing

agated to its master copy. Global commit can happen only when the network is connected. Successful commitment of a mobile transaction occurs if and only if its two sub-transactions are successfully committed.

The processing of a mobile transaction consists of two phases, denoted by $MT = T_{master} \rightarrow T_{slave}$. A master sub-transaction executes in the first phase of mobile transaction processing. Major transaction operations are performed in this phase. However, a master sub-transaction is considered tentative in that its commitment cannot make the effect of updates durable (i.e. permanent) due to either some stale data or network disconnection. A slave sub-transaction, derived from a master sub-transaction, occurs in the second phase of mobile transaction processing and its execution takes place when the data items on both the fixed part and mobile part are connected. A slave sub-transaction largely performs updates on the data items that are modified by a master sub-transaction and its commitment will make the effect of a mobile transaction durable.

In terms of two sub-systems, the place where a master sub-transaction executes is always opposite to the one where its slave sub-transaction does. The master sub-transaction of a DS transaction, or DS_{master} *sub-transaction*, is processed on a mobile

transaction host with cached data from the fixed part. The corresponding slave sub-transaction, DS_{slave} *sub-transaction*, is processed on a fixed host. Conversely, the master sub-transaction of a TS transaction, or TS_{master} *sub-transaction*, is processed on a fixed host with the replication of the data items from the mobile host. The matching slave sub-transaction, TS_{slave} *sub-transaction*, is processed on the mobile host. In this way, the effect of data modification can be propagated to its master copy.

When a mobile transaction is issued from a mobile host, its master sub-transaction is processed with cached data if network connectivity between the mobile transaction host and its local base station is unsatisfactory (i.e., poor bandwidth or disconnected). If required data is not available, the transaction is aborted; otherwise, after a tentative commit, a slave sub-transaction is issued and put into a queue. When the mobile transaction host restores its connection, the queued slave transactions are first submitted through a base station to a fixed host with a transaction co-ordinator. If a slave sub-transaction fails, the entire mobile transaction is aborted; otherwise the original transaction is committed.

If network connectivity is strong when a mobile transaction is started, the location where its master sub-transaction is processed depends on a scheduling decision. If the decision is made to use a data shipping policy, the fixed data items required by the transaction are downloaded from the FDB (the size of downloaded data is determined by the amount of valid data in the cache of the mobile transaction host). After this, the process of mobile transaction processing is similar to the one above when network connectivity is poor.

On the other hand, if the decision is made to use a transaction shipping policy, the mobile data items required by the transaction are uploaded from a mobile part owned by the mobile transaction host, to a fixed host, where the master sub-transaction is processed and a tentative commit is made. Similarly, at this time, if network connectivity between the mobile transaction host and its local base station is below some threshold, the slave sub-transaction will be queued and executed later when

network connectivity is improved. If network connectivity remains strong, the slave sub-transaction that performs updates of mobile data items on the mobile transaction host can be issued as soon as the master sub-transaction is completed.

To maintain global data consistency, each master sub-transaction has its slave sub-transaction perform the updates. Because a master sub-transaction and a slave sub-transaction is separated, the effect of data modification is propagated asynchronously. This approach is like a modified *lazy master* replication [CL91]. An analysis shows it is suitable for mobile computing environments. The data consistency issue will be further discussed in Section 3.4.

To summarise, the ATS approach has the following characteristics:

1) A mobile transaction can be tentatively committed based on locally cached data and other transactions can be submitted during disconnection so that autonomous operations can be supported;

2) A mobile transaction can be either executed on a mobile host or a fixed host based on a run-time decision;

3) The execution of a mobile transaction is two-phased. A transaction becomes durable when the sub-transaction in each phase is successfully committed.

A mobile transaction is simply a distributed transaction in a mobile environment. Some parts of the computation are performed on mobile hosts and some parts on fixed hosts [PB94]. The problem is that no general method exists to determine what part of a transaction should be executed on a mobile host and what part on a fixed host. With the ATS approach, rather than splitting a transaction into different parts, a transaction as a unit is scheduled and executed entirely on either a mobile host or a fixed host based on status information about the current computing environment.

3.3.3 Decision Objectives and Criteria

The run-time decision-making based on mobility information is the key of the ATS approach. In this section, a description of the decision problem and objectives are

given, followed by the development of detailed decision criteria.

Decision Problem and Objectives

With the ATS approach, a mobile host always submits transaction requests to the GTS if the network condition is satisfactory; otherwise it will be processed locally with cached data. A *transaction request (TR)* is a mobile transaction plus the required control information. In this work, a transaction request consists of a set of database operations (T_{OP}), a set of data items on which the operations are performed, i.e, its base set (T_{BS}), the cache status (T_{CS}) of the data items in the base set, the mobility status (T_{MS}) of the mobile host, and user specified application information (T_{AI}). Thus, a transaction request is identified by the 5-tuple $TR = (T_{OP}, T_{BS}, T_{CS}, T_{MS}, T_{AI})$.

The cache status of data items is given by a set of cache timestamps. It includes two types of cache timestamps. One is the timestamp of the fixed data items cached by the mobile host and another is the timestamp of the mobile data items that are owned by the mobile host (i.e. primary copies). Details of the cache status information are provided in Section 3.4.

The user specified application information may include the significance of data items to support autonomous operations and some tentative commit criteria when disconnection occurs. In this work, only the information about the significance of data items is considered. Each data item has an attached number that indicates a user's preference for caching the data. This information can be collected in different ways, including some statistical methods, a system default, or individual user profiles. The assumption is that the greater the value, the higher priority for caching. The average preference of data items in the base set of a mobile transaction is defined as a *cache priority* of the mobile transaction, denoted by P_i.

The mobility status T_{MS} is determined by several mobility factors. Its formal definition is presented in Chapter 2. T_{MS} could be represented as a vector or a single value. In the following work, the vector representation is used. Regarding

the mobility factors, of particular interest in this work is the network bandwidth, disconnection ratios, and residual battery life of a mobile host. In general, let T_i be a mobile transaction request submitted by a mobile host h_i ($i = 1, 2, \ldots, m$) at a particular time point. Let the tuple $< B_i, D_i, E_i >$ be the mobile status vector at this time, where B_i, D_i and E_i are the bandwidth, disconnection ratio, and residual battery life of the mobile host, respectively. With these parameters, the decision problem is stated as follows: given a mobile transaction request, how should its execution should be scheduled? More generally, given a set of mobile transaction requests, can this set be partitioned into DS transactions and TS transactions so that better execution strategy is employed.

The solution to this problem depends on the desired decision objectives. In this research, they are stated as follows:

1) support independent autonomous computation on a mobile host during its disconnection; and

2) have maximum system throughput and better response time.

The first objective simply means that a mobile host should be allowed to continue its transaction processing during disconnection. Thus, required data should be transferred to the host and cached so that the mobile host can use it after disconnection. The second objective implies that the overall system performance should be balanced. Overall system performance should not be sacrificed to support autonomous operations of a single mobile host. For instance, if a pessimistic locking scheme is used autonomous operations can be performed for a disconnected mobile host, but it is not a good strategy because the locked data items may not be available for a long time.

The transaction processing during a disconnection can only use local resources of a mobile host. However, in a normal condition the transaction processing should make better use of the system resources available on a fixed network. This means that if the network connection is stable, transaction processing should be similar to

the traditional approach. When the network connection is poor, processing should be locally dominated. As we can see, these two objectives depend much on the computing environment of a mobile host. Therefore, a dynamic and adaptive run-time assessment is necessary.

Decision Criteria

Bearing in mind the objectives introduced above, the decision criteria to select a transaction processing policy for a given mobile transaction T_i is specified in this section. To simplify the description, the notations used in the following section are first introduced. Given a T_i (accordingly a transaction request TR_i), let T_{i_d} and T_{i_u} denote the time of downloading data and the time of uploading data, respectively. Let T_{i_p} be the patience time of a mobile user for that transaction. Moreover, let B_i and E_i be the current bandwidth and the residual battery life. The $schedule(T_i, x)$ is used to represent scheduling T_i as an x transaction, where $x \in \{DS, TS\}$.

The basic idea of selecting a transaction processing policy is that given a mobile transaction request, if the computing condition is normal (strong connectivity and no special application or resource constraints), it should be treated in the same manner as a conventional environment. That is, submit the transaction to a fixed host and let it run from there. If the computing environment suffers from low bandwidth that is below a specified level, then the transaction will be processed locally by the mobile host.

With the information provided by a transaction request such as mobile status, cache status and user specified application information, two decision criteria are as follows. First, if the residual battery energy is below a specified level, then the transaction should be executed on a fixed host. Second, if the network bandwidth is weak, i.e. below a specified level, it is treated like a disconnection and so the mobile transaction request will not go to the GTS and instead is scheduled as a DS transaction. These two simple cases can be expressed in the following heuristic rules:

(1) IF $E_i < H_e$ THEN $schedule(T_i, TS)$

(2) IF $B_i < H_b$ THEN $schedule(T_i, DS)$

Other criteria to determine where to execute the transaction can depend on the amount of required data transmission as well as the user's application requirements. When the network bandwidth is above a specified level, regardless of application requirements of a mobile user, the decision will be made based on the time of downloading/uploading, which is the function of the size of downloading/uploading and current network bandwidth. Upon the receipt of a transaction request, the GTS can determine the amount of data to download/upload from the cache status information. This amount gives the size of stale data cached on the mobile/fixed host. If cached data for a current transaction is valid, no data will be transported. Therefore, if the size of downloading data from the fixed part to the mobile host is smaller than the size of uploading data from the mobile host to the fixed part, then the transaction is scheduled as a DS transaction; otherwise it is scheduled as a TS transaction. This is expressed as the following rule:

(3) IF $T_{i_d} < T_{i_u}$ THEN $schedule(T_i, DS)$ ELSE $schedule(T_i, TS)$

Example 3.1:

Suppose a given mobile transaction request is required to download 50K if it is processed on the mobile host and to upload 35K if it is processed on the fixed host. At the time of transaction submission, the network bandwidth is 32Kbps. Then the time of downloading/uploading data is 12.50/8.75 seconds, respectively. Because the uploading data takes a shorter time than downloading , a transaction request may be scheduled as a TS transaction.

The assessment based on the time of downloading/uploading data is just one piece of the overall picture. In some cases, even though the time of downloading data is longer than uploading, a mobile user wants to transfer the data to the mobile host because the data is required during disconnection, especially for predictable disconnection. This application requirement is reflected by a *patience time* of a

mobile user. The patience time from the viewpoint of a mobile user indicates the significance of required data items and the data transmission delay it can tolerate. Before giving the 4th decision rule, the concept of patience time is first discussed.

Modelling user patience in mobile environments is proposed in the Coda system [MES95]. The purpose is to improve usability by reducing the frequency of user interaction. User patience refers to the amount of time a mobile user is willing to spend receiving required data before it becomes disconnected. In [MES95], a model to estimate user patience τ is suggested, i.e.

$$\tau = \alpha_1 + \alpha_2 e^{\alpha_3 X}$$

In the formula, $\alpha_i (i = 1, 2, 3)$ are scaling factors, and X is an integer ranging from 0 to 1000, which describes the importance of a data item from the user's perspective.

In this work, the user patience model is extended. Disconnection factor is incorporated as an additional component of the formula. Moreover, instead of measuring the importance of a single data item, the cache priority is used to reflect the overall significance of all the data items appearing in a mobile transaction.

Observation and intuition shows that with more frequent disconnection of a mobile host, the user displays less patience. Therefore, a modified formula for the patience of a mobile host is given as follows:

$$\tau = \alpha_1 + \alpha_2 (1 - D) e^{\alpha_3 P}$$

In the extended formula, P stands for the cache priority and D for the disconnection ratio (the definitions for these are given in the Section 2.6). One extreme case of the formula is that if the disconnection ratio of the mobile host is 0, the patience time is entirely determined by the cache priority which is the same as the original form. Conversely, if the disconnection ratio is 1, the patience time becomes a constant (α_1), i.e. a minimum patience time.

Example 3.2:

Following the Example 3.1, suppose the required data of the mobile transaction involves four data items. Let their individual cache preferences be 500, 400, 650, and 200, respectively. Thus, the cache priority is (500+400+650+200)/4, approximately 438. Moreover, let the disconnection ratio of the mobile host in any given time interval be 0.45 and the scaling factor $\alpha_1 = 2, \alpha_2 = 1$ and $\alpha_3 = 0.01$. Then the estimated patience time of the mobile host with the extended formula is given as follows:

$$\tau = 2 + 1 \cdot (1 - 4.5)e^{0.01 \cdot 438} = 2 + 5.5 \cdot e^{4.38} = 45.69(seconds).$$

Thus, given a mobile transaction request, if the patience time is larger than the time of data downloading, then the transaction is scheduled as a DS transaction. That becomes the 4th decision rule.

(4) IF $T_{i_p} > T_{i_d}$ THEN $schedule(T_i, DS)$

Example 3.3:

Following the Example 3.1, based on user specified application information in the transaction request, suppose the patience time of the user is 45.69 seconds. Then the transaction will be scheduled as a DS transaction because the data items used in the current transaction is significant (with a large cache priority) to support autonomous operations during the next disconnection.

Combining these two cases together, we say that a processing policy of a mobile transaction is first determined by user's patience time, then it is determined by the time of data transmission. However, in the latter case, the patience time will be used to determine whether the data will be really transferred or not. For instance, if a transaction is scheduled as a TS transaction based on the time of downloading/uploading data, but the patience time is smaller than the time of uploading data, then the data is not physically moved due to short patience time. Therefore, the cached data will be used.

Example 3.4:

Following the Example 3.1, based on user specified application information in the transaction request, suppose the patience time of the user is 6.60 seconds. Then the transaction will be scheduled as a TS transaction because data items used in the current transaction is not significant (with a small cache priority) to support autonomous operations, and the time of uploading data is less than the time of downloading. However, since the patience time is shorter than the time of uploading data, no data will be physically moved from the mobile host to a fixed host. Instead, cached data will be used.

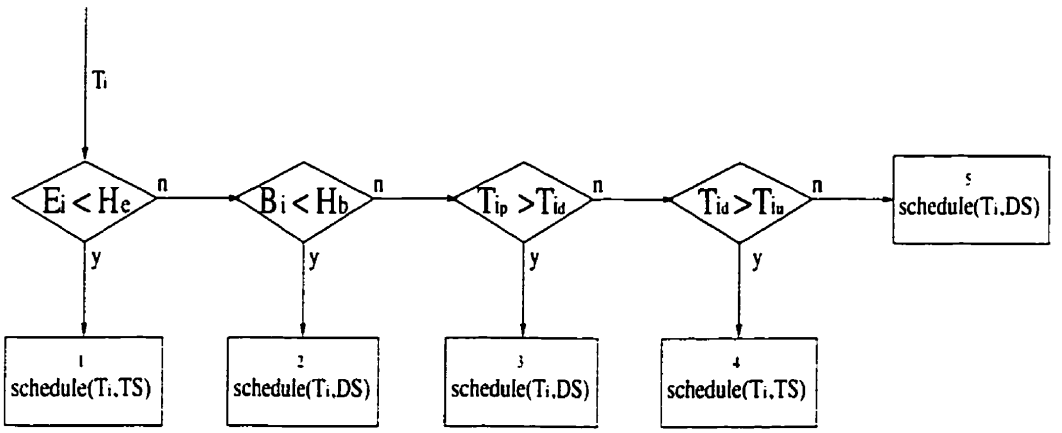


Figure 3.7: The decision flow of mobile transaction scheduling

In summary, the above description can be expressed as a decision flow, shown in Figure 3.7. In the figure, H_e and H_b are two thresholds and indicate satisfactory battery and bandwidth levels, respectively. They can be specified at the time when a mobile host registers, and can be modified by a mobile user through a negotiation with the system.

3.4 Cache Consistency Control

The support of disconnection, bandwidth savings, and reduced response time are major benefits of executing a mobile transaction on a mobile host. However, availability

is a trade-off for consistency. Since the network may become partitioned, the control of data consistency becomes more complicated and more difficult.

With the ATS approach in this work, a GDB is fundamentally separated into two parts: fixed part and mobile part. Accordingly, the mobile transactions are separated into two groups: DS transactions and TS transactions. When a network partition occurs, the transactions in each group can be processed independent of one another with cached data. However, in the previous description of the ATS approach, the data consistency between a cached copy and its primary copy is not fully discussed. This issue is addressed in this section.

Cache consistency control in the ATS approach is largely based on a time certification based scheme that is proposed in [Nie95]. With this scheme, it is assumed that a data item is cached in a page. A cache status is expressed as a tuple with seven elements: $(DN, P\#, UT, CT, CF, CTB, UTB)$. The first two elements, data item name (DN) and page number $(P\#)$ uniquely identifies a cache unit (i.e. a page) of a data item. The elements (UT) and (CT) stand for the latest modification time of the primary copy of a data item and the time when a cached copy is invalidated, respectively. The fifth element (CF) is a consistency flag to indicate if the cached copy is consistent with its primary copy at the time of comparison. The last two elements, (CTB) and (UTB), are consistency time bound and modification time bound, respectively. These two elements are user controllable and reflect the degree of consistency requirements. For instance, a large CTB value (relative to the CT) represents a more optimistic view of consistency than a small value, whereas the smallest (identical to the CT) represents strict consistency.

Regarding cache invalidation, the basic idea of the scheme is that if the CT of a cached copy is within the CTB bound, the cached data is considered valid. Similarly, regarding cache modification, the UTB indicates the maximum time for a cached copy to be temporarily modified. Beyond this time, local modification of a cached copy must be reflected on its primary copy. This requires the establishment of a network connection and a write to the primary copy. At this time, if the network connection

fails, the effect of the write must be reversed to a previously consistent state.

In addition to the time points for the cache invalidation and modification indicated by CTB and UTB, the ATS approach can verify the cache consistency at the submission time of a transaction request. At this time, the network connection is established. The cached copy and primary copy of a data item is made consistent. If the verification of cache status takes place only at the submission time of a transaction request, this becomes a highly optimistic scheme. A highly optimistic scheme assumes that cached data is valid during the time between two transaction requests. This scheme may result in a high chance of abort if the time interval between two transaction requests is large or the modification of primary copies is frequent. Since a mobile computing environment is varying, it is desirable to have consistency control be able to adapt to the environment and application requirements. Hence, using the time bound parameters CTB and UTB to control cache consistency is necessary. For instance, if network condition is always strong, both parameters can be set to zero so that a strict consistency is maintained; otherwise weak consistency criterion may be applied based on network condition.

The effectiveness of the time certification based scheme depends on the correct estimation of CTB and UTB. They rely on the access pattern to a primary copy of data, the nature of an application, and the network conditions. How to determine these two parameters is an open question. In fact, it is possible to take mobility information into account to determine the time bound dynamically. However, this is open for future investigation.

3.5 An Illustrative Example

An example is presented in this section to illustrate the process of transaction processing with the ATS approach. To simplify the description, it is assumed that only one mobile host is in the system and the bandwidth is above a certain threshold. The system consists of three cells C_1,C_2, and C_3. Accordingly, the system has three base stations. Each one is responsible for a cell.

Suppose the mobile host issues four transactions T_1,T_2,T_3, and T_4 from cell C_2,C_3,C_2, and C_1, respectively, when the mobile host is moving. (Note that the notation T_i here varies slightly from that used in Section 3.3.3). The battery power level and the bandwidth are assumed to be over the specified thresholds when the transactions are submitted. Figure 3.8 shows the system architecture in this example. Table 3.1 lists the information derived from a transaction request. For instance, at the time when the first transaction request is submitted, the disconnection ratio (D), the bandwidth (B) and the cache priority (P) are 0.1, 9.6Kbps, and 250, respectively. Based on the cache status, the size of required data to download (R_d) and upload (R_u) are 100k and 50k, respectively.

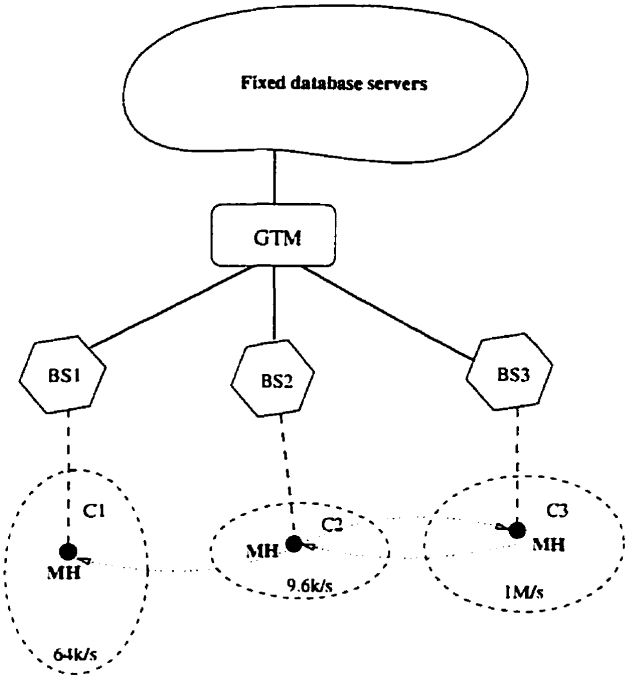


Figure 3.8: The system architecture in the example

Using the decision rule developed in Section 3.3.3, we have the result as shown in Table 3.2. It shows that the transaction T_1 and T_3 are scheduled as TS transactions and T_2 and T_4 are scheduled as DS transactions.

For the transaction requests T_1 and T_3, because the application requirements of the user are not dominating, the decision (i.e., box 4 in Figure 3.7) is made based on

Table 3.1: The decision information of transaction requests

No.	D	B	P	R_d	R_u
T_1	0.1	9.6Kbps	250	100k	50k
T_2	0.6	1Mbps	200	50k	30k
T_3	0.2	9.6Kbps	450	95k	40k
T_4	0.4	64Kbps	50	30k	35k

Table 3.2: The scheduling result of transaction requests

No.	T_d	T_u	T_p	*decision*
T_1	83.33s	41.67s	12.96s	TS
T_2	0.40s	0.24s	4.96s	DS
T_3	79.17s	33.33s	74.01s	TS
T_4	3.75s	4.38s	2.99s	DS

the size of data to download/upload and the bandwidth at that time. Nevertheless, for the first transaction request, since the patience time is shorter than the time of data uploading, the data transmission may not physically take place. In this case, the replica on the fixed host is used. For the third transaction request, the data is transported to the fixed host because the transport time is sufficient.

For the second transaction request, because the patience time of the user is higher than the time for downloading data, it is scheduled as a DS transaction and the required data is physically moved to the mobile host. The fourth transaction is also scheduled as a DS transaction. However, the difference between these two decisions is that the second one is scheduled due to a higher patience time (see box 3 in Figure 3.7), whereas the fourth one is due to a small amount of data to download (see box 5 in Figure 3.7).

If the data is not highly desired for autonomous operations, the communication cost is always lowest with the ATS approach. However, in general, this does not hold because supporting autonomous operations may require a longer time be spent downloading data. Nevertheless, the price paid for this is expected to enhance the

overall system performance in terms of throughput and battery energy consumption provided frequent disconnection exists.

The above example is a simplified illustration of how a decision is made to schedule a particular transaction request. In this example, in terms of the time of data transmission, if the fixed transaction shipping (FTS) policy is used, the four transactions take 79.62 seconds. If the fixed data shipping (FDS) policy is used, they take 116.65 seconds. With the adaptive shipping (AS) policy, they take 79.15 seconds.

For this particular case, the performance improvement with the ATS approach is not significant. This is not surprising because the ATS approach does not guarantee the best in all the circumstances. In fact, the ATS approach seeks for the trade-off of overall system performance. Its performance depends on the network connectivity and transaction condition. When the network connectivity deteriorates, the ATS becomes superior to the other two approaches. This claim is proved by the simulation experiments. The simulation model and results are described in the next chapter.

Chapter 4

SIMULATION MODEL AND EXPERIMENT

The performance of the ATS approach is evaluated by the means of simulation. In this chapter, the design and implementation of a simulation model are described. Further, the results of the simulation experiments are reported and discussed.

4.1 Simulation Objectives

The goals of this simulation are twofold. First, it explores the performance of the ATS transaction approach with mobility information relative to the two fixed transaction processing approaches (i.e. FDS and FTS). Second, it reveals the effect of network connectivity on transaction processing performance. More specifically, the following six measurements are gathered and analysed:

1. The total response time to process a given number of transactions for each approach under a given inter-event time of bandwidth changes with different disconnection ratios;

2. The total response time to process a given number of transactions for each approach under a given disconnection ratio with varying inter-event time of bandwidth changes;

3. The total processing time on a mobile host to serve a given number of transactions for each approach under a given inter-event time of bandwidth changes with different disconnection ratios;

4. The total processing time on a mobile host to serve a given number of transactions for each approach under a given disconnection ratio with varying bandwidth changes;

5. The system throughput during a given time interval for each approach under a given inter-event time of bandwidth changes with different disconnection ratios;

6. The system throughput during a given time interval for each approach under a given disconnection ratio with different inter-event times of bandwidth changes.

These six performance measures are designed for different purposes. The first two aim at comparing the response time among the three approaches. The third and fourth measure the time spent on a mobile host and can be used to estimate the energy consumption of different approaches. The last two investigate the performance based on system throughput. The experiments are therefore designed to reveal the effect of the bandwidth change and network disconnection on response time, mobile host's processing time, and system throughput.

4.2 Model Structure Description

This simulation experiment focuses on a single mobile host in a cell. The mobile system is abstracted as a queuing model. Its general structure is shown in Figure 4.1. The model consists of three servers (in the terminology of queuing theory) labelled S_B, S_M, and S_F. The server S_B stands for the GTS related services. Normally, a mobile transaction request gets this service first. A mobile transaction is scheduled as a DS or TS transaction based on available mobility information. The server S_M represents the services provided by a mobile transaction host and the server S_F the services provided by a fixed host where transaction co-ordination takes place. In

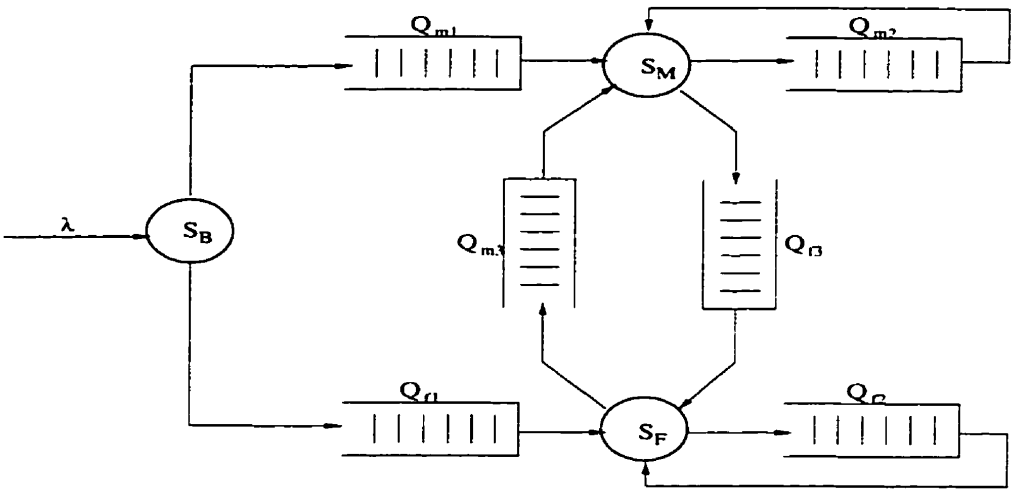


Figure 4.1: The queuing model overview of ATS approach

general, the S_M and S_F each have three queues. If a transaction is scheduled as a DS transaction, it goes into queue Q_{m1}; otherwise into Q_{f1}.

According to the ATS approach, for a DS transaction, after its master sub-transaction is processed, its slave sub-transaction is put into the queue Q_{m2}. If the network is connected, a DS slave sub-transaction is submitted to a fixed host for processing and enters Q_{f3}; otherwise it has to wait in Q_{m2} for network re-connection. If a DS slave sub-transaction is successfully committed, the original mobile transaction is completed; otherwise, the transaction fails. After a DS slave sub-transaction is finished, the entire transaction is completed. For a TS transaction, this process is identical, but with the different notations of Figure 4.1.

The ATS approach can be viewed as a combination of the two basic approaches FDS and FTS. For the FDS approach, a mobile transaction goes through four processing steps: data downloading, master sub-transaction processing, slave sub-transaction submission, and slave sub-transaction processing. Thus, there are four simple services.

Similarly, the FTS approach has four simple services as well. The difference between the FTS model and FDS model is that the data required by a mobile transaction is uploaded from the mobile host to a fixed host. Besides, the first three simple services, i.e. data uploading, master sub-transaction processing and slave

sub-transaction submission, are implemented on a fixed host, whereas slave sub-transaction processing is performed by the mobile host that issues the transaction.

Transaction scheduling (i.e. S_B) has no elapsed service time and physically such a service cannot be reached during disconnection. Technically, it is not suitable to considering it as a service centre. However, in this simulation model, the GTS processing is immediately followed by data downloading/uploading. Therefore, the GTS service and data transmission are combined together and treated as a single service centre.

According to the ATS approach, when a mobile host is disconnected a transaction is treated as a DS transaction and has no data transmission. If the network is connected, the data will be uploaded or downloaded based on current scheduling decision. Therefore, the S_B is included in the simulation model and it is assumed that this service is always available.

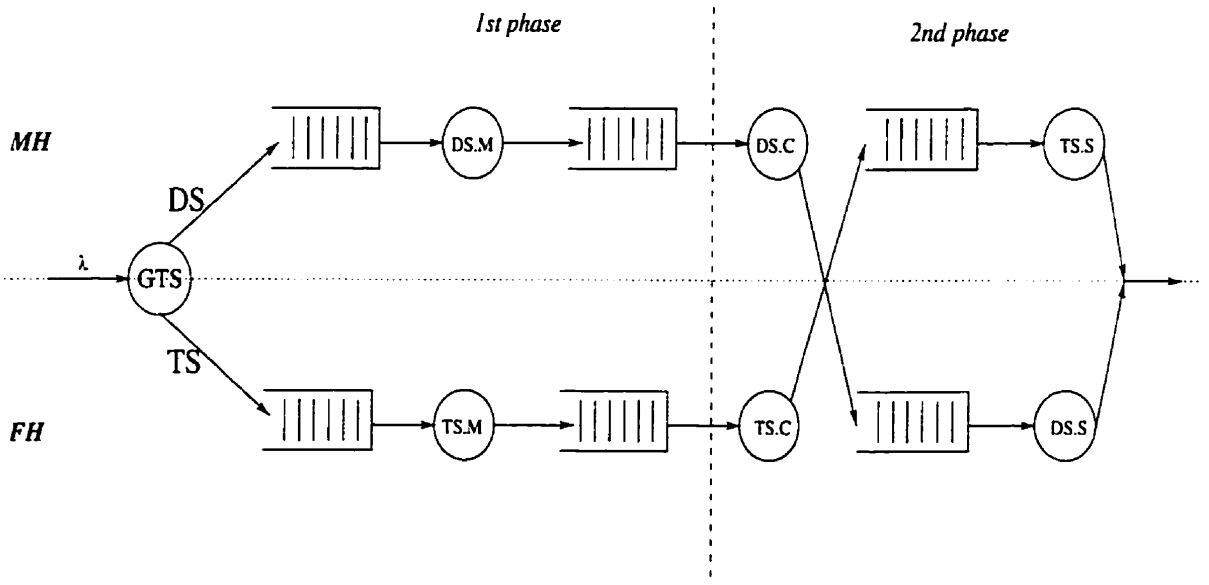


Figure 4.2: A specific queuing model of ATS approach.

Figure 4.2 illustrates the model structure by splitting the service centres into simple service centres. In Figure 4.2 the GTS provides transaction scheduling plus data transmission service. The $DS.M/TS.M$, $DS.C/TS.C$, and $DS.S/TS.S$ provide master processing, slave sub-transaction communication, and slave sub-transaction

processing services for DS / TS transactions, respectively.

It can be seen that the GTS links the two basic models together and the entire model has a symmetric structure. The upper part of the model is for a mobile host and the lower part is for a fixed part. The execution of a mobile transaction goes through from the first phase (the left part) to the second one (the right part).

4.3 Experiment Assumptions

Performing experiments with the above general model can be complex without certain restrictions on transaction submission. If transaction requests cannot be processed quickly, they wait in one of the queues. For read-only transactions this is not a problem. However, if update transactions are allowed, a transaction failure can result in cascade abort of the following transactions. This requires extra modelling effort to reflect this reality. Since this simulation explores the relative performance of various approaches, the model is simplified with the following assumptions:

1. In the normal condition of a fully connected network a transaction request from a mobile host is not issued until the previous one is completed. If a transaction is blocked due to disconnection, or has completed, a new transaction request is submitted.

2. Cascade abort is not modelled. That is, we assume that a transaction abort does not affect the subsequent transactions.

3. A mobile transaction performs a fixed number of operations on a fixed number of data items and each operation takes a fixed amount of time.

4. The arrival of network bandwidth change is subject to a Poisson distribution.

5. The messages about success or failure of a slave sub-transaction are assumed to be transmitted in broadcast mode, and the effect of such message processing delay is ignored.

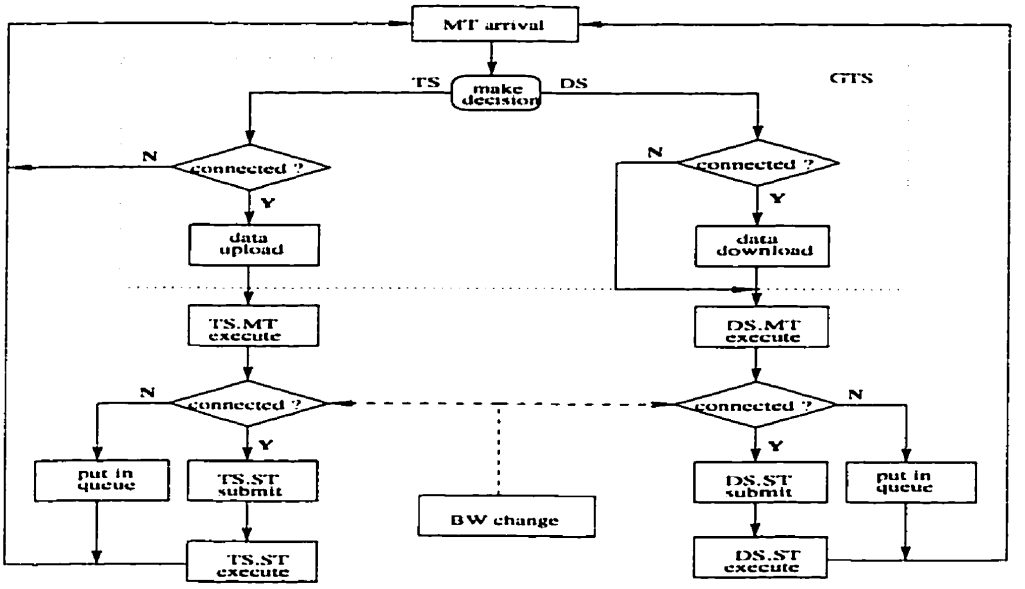


Figure 4.3: The simulation logic of ATS approach

The simulation logic of the ATS approach is specified in Figure 4.3. The figure shows that transactions queue up only after the execution of a master sub-transaction. In fact, the first assumption ensures that the four queues: Q_{m1},Q_{m3},Q_{f1}, and Q_{f3} in Figure 4.2 can be eliminated. The reason for this is clear if the network is fully connected only one transaction is submitted and processed at a time so no queues are necessary. However, if the network is not fully connected, the queues Q_{m2} and Q_{f2} are necessary even if only one transaction request is allowed to enter the system. Network bandwidth change takes place randomly or in a specified pattern. Once the bandwidth is above a pre-defined threshold, a transaction request is dequeued before another can be submitted.

Generally, the simulation is conducted for the three policies separately. If the policy is FTS, transaction requests are always scheduled as TS transactions. If the network is disconnected when a request arrives, this request must give up because its first phase cannot be carried out. After the completion of the first phase, if the network is not connected at this time, the slave sub-transaction cannot be submitted and must wait in the queue. Unlike the FTS approach, the FDS policy usually does not discard a transaction request. Instead, the first phase of transaction processing

is performed by using cached data, even if the network is disconnected. However, if the network is disconnected after this phase, the request must be queued for later processing. For the ATS policy, when the network is disconnected, it functions in the FDS approach; otherwise, it functions as a mixture of the FTS and FDS.

According to the network connection states, the data may or may not be transported during the first phase of a DS transaction. Moreover, a slave sub-transaction may or may not be blocked. The network connectivity directly impacts the performance of transaction processing. To reflect this, the time spent on the commitment of a DS transaction can be distinguished into four cases. These will be specified in Section 4.4.4.

4.4 Model Parameters

In this work the discrete event-driven simulation technique is used. The parameters that characterise the simulation model are distinguished into three categories: *event parameters*, *service parameters*, and *performance parameters*. Their notations and meaning are presented in this section. The functions used to determine the service times in the simulation are discussed as well.

4.4.1 Event Parameters

The change of network bandwidth, transaction data communication, master sub-transaction execution, slave sub-transaction submission and execution are modelled as events. Among them, according to the assumptions, only bandwidth change is an independent event. The other events are triggered by related events. The event parameters of interest are listed in Table 4.1.

The mean inter-event time (*mean.BW.change*) of bandwidth change is a random variable and subject to exponential distribution. For the simulation, the whole range of bandwidth is separated into a number of intervals, and each interval has a fixed bandwidth (*BW_level.ITV*). At a particular time, the bandwidth is specified by

Table 4.1: Major event parameters of simulation model

No.	event param.	Note
1	*mean.BW.change*	average inter-event time of bandwidth change (random)
2	*BW_level*	the number that indicates current bandwidth level
3	*BW_level.ITV*	the bandwidth interval
4	*BW.threshold*	pre-defined bandwidth threshold
5	*MT.DS.DI.number*	the number of data items accessed by a DS transaction (random)
6	*MT.DS.OP.number*	the number of operations of a DS transaction (random)
7	*MT.DS.ST.size*	the size of data for a DS slave sub-transaction (random)
8	*MT.TS.DI.number*	the number of data items accessed by a TS transaction (random)
9	*MT.TS.OP.number*	the number of operations of a TS transaction (random)
10	*MT.TS.ST.size*	the size of data for a TS slave sub-transaction (random)
11	*MT.CP*	cache priority (random)
12	*MT.type*	transaction type
13	*policy*	transaction scheduling policy

(*BW_level*). For instance, let the maximum bandwidth be 1000Kbps and let it be separated into 10 intervals. Then *BW_level.ITV* will be 100Kbps. If *BW_level* = 3, it means that current network bandwidth is $3 \times 100Kbps = 300Kbps$. In the simulation, a parameter *BW.threshold* is used to indicate a satisfactory bandwidth level from the perspective of applications. If the current bandwidth is below this threshold, the network is treated as disconnected.

As mentioned in Section 4.3, a transaction is assumed to access a number of data items and perform some operations. These parameters are represented in transactions by *MT.xx.DI.number* and *MT.xx.OP.number* (where xx can be DS or TS), respectively. They are treated as random variables that are subject to a uniform distribution. The size of a slave sub-transaction is denoted by *MT.xx.ST.size*. The cache priority (*MT.CP*) for a given transaction is treated as a random variable as well and assumed to be subject to normal distribution. Besides, in this simulation, the disconnection ratio of a mobile host is determined by the average disconnection duration over the simulation time interval. The disconnection duration is controlled by adjusting the bandwidth threshold (*BW.threshold*).

A mobile transaction is modelled as a temporary entity and distinguished into four

types (denoted by *MT.type*): CC,CD,DC, and DD, based on the network connectivity in the life cycle of a transaction execution. A CC-type transaction indicates that the network is connected at the time of the data transmission and the time of slave sub-transaction submission. A CC-type transaction is just like a normal transaction that the network is in a fully connected condition. The other three types of transactions indicate that at least one of two communication steps is broken (denoted by the D). The performance of these four types of transaction can be different. The functions to determine the execution time will be discussed in the next sub-section.

To perform experiments on different approaches, the model introduces a parameter (*policy*). This parameter is simply a constant (1 for FDS,2 for FTS, and 3 for ATS) that allows the simulation to switch from one approach to another.

4.4.2 Service Parameters

Table 4.2: Major service parameters of simulation model

No.	service param.	Note
1	data.dnloading.time	data downloading time
2	DS.MT.exe.time	execution time of a DS master sub-transaction
3	DS.ST.com.time	communication time of a DS slave sub-transaction
4	DS.ST.exe.time	execution time of a TS slave sub-transaction
5	data.uploading.time	data uploading time
6	TS.MT.exe.time	execution time of a TS master sub-transaction
7	TS.ST.com.time	communication time of a TS slave sub-transaction
8	TS.ST.exe.time	execution time of a DS slave sub-transaction

There are seven servers in the simulation model. Each one has its own service time. The parameters representing these service time are listed in Table 4.2. The data downloading or uploading is a mutually exclusive operation performed based on the decision made by the GTS server. For a transaction scheduled as a DS transaction, the execution time of the master sub-transaction is denoted by *DS.MT.exe.time*; the communication time spent on submitting a slave sub-transaction is denoted by *DS.ST.com.time*; and the *DS.ST.execution.time* represents the execution time of a slave sub-transaction including the final commitment time of a mobile transaction.

There are similar parameters for a transaction that is scheduled as a TS transaction, but with TS as the prefix. The functions used to determine these service times are explained in Section 4.4.4.

4.4.3 Performance Parameters

According to the experiment goals, the response time (*RP.time*), the service time of each type of transaction (*DS.service.time*, *TS.service.time*) and throughputs of the system (*DS.number*,*TS.number*) at a given time interval (*close.time*) are considered as the performance parameters monitored in the simulation. These parameters are listed in Table 4.3.

Table 4.3: Major performance parameters of simulation model

No.	performance param.	Note
1	*RP.time*	response time
2	*DS.number*	the number of scheduled as DS transaction
3	*TS.number*	the number of scheduled as TS transaction
3	*DS.service.time*	the time spent on DS transaction processing on a MH
4	*TS.service.time*	the time spent on TS transaction processing on a MH
6	*close.time*	time duration of simulation

The parameter *RP.time* records the total response time (from transaction submision to commitment) for a given number of transactions. In each experiment, the number of transactions scheduled as DS transactions and the number of ones scheduled as TS transaction are given by *DS.number* and *TS.number*, respectively. When simulation time (*close.time*) is specified, the total number of transactions (*DS.number* + *TS.number*) is used to show the system throughput. The *DS.service.time* and *TS.service.time* give the time spent on DS and TS transaction processing on a mobile host. These two parameters are used as an estimation of battery consumption for each approach.

4.4.4 Service Time Functions

Corresponding to the eight parameters listed in Section 4.4.2, Table 4.4 gives the formulae used to determine the service time of the servers in the simulation model.

Table 4.4: The functions that determine service time

No.	service time	function
1	data.dnloading	c+(MT.DN.size × 8)/(BW_level(MH) × BW_level.ITV)
2	DS.MT.exe.time	MT.DS.OP.number × MT.DS.DI.number × UNIT.OP.time
3	DS.ST.com.time	c+(MT.DS.ST.size × 8)/(BW_level(MH) × BW_level.ITV)
4	DS.ST.exe.time	MT.DS.ST.size × UNIT.OP.time + CC.COMMIT.time)
	DS.ST.exe.time	MT.DS.ST.size × UNIT.OP.time + CD.COMMIT.time)
	DS.ST.exe.time	MT.DS.ST.size × UNIT.OP.time + DC.COMMIT.time)
	DS.ST.exe.time	MT.DS.ST.size × UNIT.OP.time + DD.COMMIT.time)
5	data.uploading	c+(MT.UP.size × 8)/(BW_level(MH) × BW_level.ITV)
6	TS.MT.exe.time	MT.TS.OP.number × MT.TS.DI.number × UNIT.OP.time
7	TS.ST.com.time	c+(MT.TS.ST.size × 8)/(BW_level(MH) × BW_level.ITV)
8	TS.ST.exe.time	MT.TS.ST.size × UNIT.OP.time + CC.COMMIT.time

The service time of a service centre is basically determined by its service type and the amount of involved data. There are roughly two types of services. One is communication oriented (i.e. No. 1,3,5 and 7 in Table 4.4) and the other is processing oriented (i.e. No. 2,4,6 and 8 in Table 4.4). For a communication oriented service, the service time consists of a constant communication overhead (c) and the data transport time, which is determined by the size of data and the bandwidth at the time point when the data is transmitted. In the formulae the $MT.DN.size$ and $MT.UP.size$ are the amount of data to be downloaded or uploaded, respectively. The $MT.DS.ST.size$ and $MT.TS.ST.size$ are the amount of data to be transmitted for a DS or TS slave sub-transaction, respectively. Taking the first formula as an example, let $c = 10$ and $MT.DN.size = 240K$. Suppose at a given time $BW\_level = 4$ and $BW\_level.ITV = 100Kbps$. Then the service time for data downloading becomes

c+(MT.DN.size × 8)/(BW_level(MH) × BW_level.ITV

=10+(240 × 8)/(4 × 100)

=14.8 (seconds).

For a processing oriented service, the service time is basically determined by the number of accessed data items ($MT.xx.OP.number$) and the average number of operations on these items ($MT.xx.DI.number$). It is assumed that each operation takes a fixed amount of time ($UNIT.OP.time$). A typical example is the service time for a DS master sub-transaction, i.e.,

$$DS.MT.exe.time = MT.DS.OP.number \times MT.DS.DI.number \times UNIT.OP.time.$$

The function to determine the service time of a TS master sub-transaction execution has a similar structure. However, to determine the service times of sub-transaction processing extra factors must be taken into account because such services depend on network connection conditions. Generally, the longer a transaction processing is delayed due to network partition, the more time it will take for the reconciliation at a later stage. As mentioned in Section 4.4.1, the DS transactions are distinguished into four types. Each type has its own service time to execute a slave sub-transaction. Therefore, the service time for a DS slave sub-transaction ($DS.ST.exe.time$) in Table 4.4 has four forms. Each one has its own $COMMIT.time$ to reflect the effect of the network connectivity.

4.5 Experiment Results and Discussion

The simulation model is implemented in SIMSCRIPT II, a general-purpose simulation language [KMV87], on Sparc/Unix platform. A few typical results are illustrated and discussed here, though many experiments with different control parameters were conducted. The experiment results are presented in four groups in this section. Each group includes three figures that has a relatively uniform investigation goal. Moreover, each figure is accompanied with a table that shows the data used for the figure and other relevant information.

The 1st Group Experiments and Results

Figures 4.4 to 4.6 are in the first group and show the impact of network condition (bandwidth) change on performance (i.e. response time) of transaction processing under a given disconnection ratio. In each experiment, 1000 transaction requests are issued. The mean inter-event time of bandwidth change varies from 0 to 100 seconds. The figures in this group give the results when disconnection ratio goes from low to high.

Table 4.5: The data for Figure 4.4 (BW.threshold=0 transactions=1000)

	FDS				FTS				ATS			
BW.change	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
10	1518.93	18.81	1000	0	475.39	15.82	0	1000	598.18	26.39	394	606
20	1553.3	28.83	1000	0	484.07	8.27	0	1000	601.85	12.31	383	617
30	1519.79	20.3	1000	0	469.43	10.76	0	1000	581.87	20.87	353	647
40	1529.1	48.27	1000	0	478.32	9.48	0	1000	606.25	19.52	398	602
50	1514.97	35.48	1000	0	477.33	7.04	0	1000	592.09	9.46	394	606
60	1505.59	40.5	1000	0	476.45	8.94	0	1000	603.97	15.68	412	588
70	1506.28	40.51	1000	0	479.85	14.53	0	1000	615.22	11.77	405	595
80	1551.87	19.14	1000	0	479.27	3.67	0	1000	601.11	12.27	388	612
90	1539.18	23.03	1000	0	469.56	3.45	0	1000	583.19	4.22	366	634
100	1499.11	29.94	1000	0	473.44	8.87	0	1000	595.06	22.03	366	634

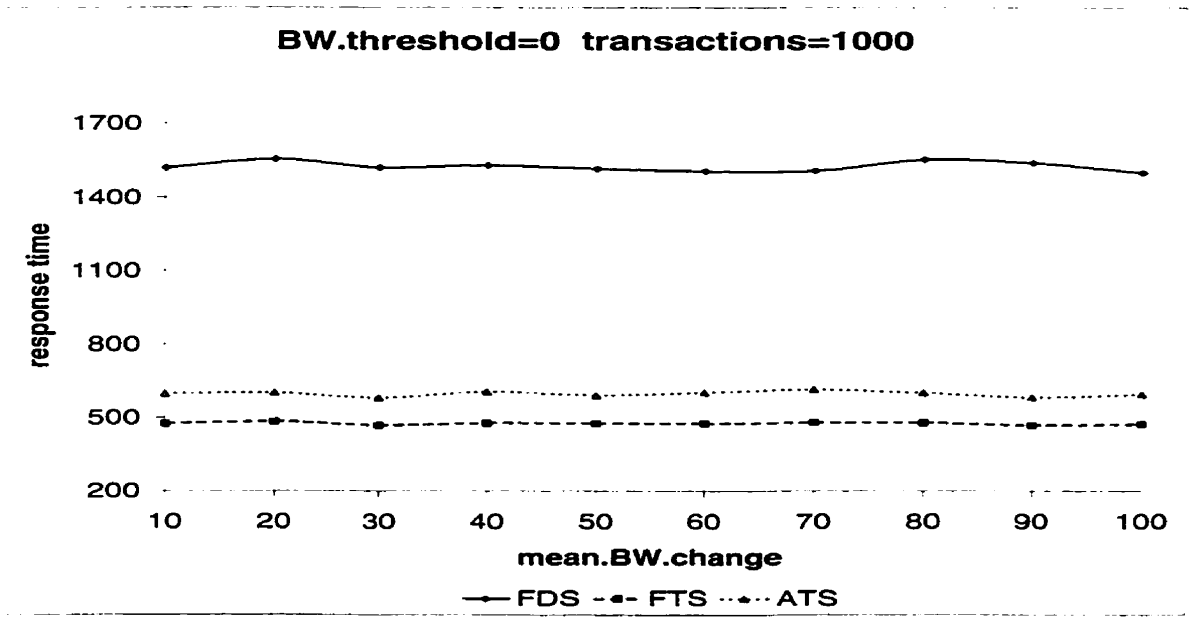


Figure 4.4: Response time vs. mean inter-event time of bandwidth change

Figure 4.4 shows the results when a mobile host has no disconnection (BW.threshold = 0). It can be seen that among three approaches, the FTS demonstrates the best performance; whereas the FDS is the worst. This is largely because generally a DS

Table 4.6: The data for Figure 4.5 (BW.threshold=5 transactions=1000)

	FDS				FTS				ATS			
BW change	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
10	2038.53	18.89	1000	0	1941.53	41.27	0	1000	1735.93	35	611	389
20	1907.59	58.34	1000	0	1954.62	20.42	0	1000	1759.08	63.34	652	348
30	2006.19	51.92	1000	0	1934.94	33.89	0	1000	1744.88	46.7	677	323
40	1971.4	49.4	1000	0	2012.64	42.54	0	1000	1759.74	20.62	641	359
50	1990.72	16.14	1000	0	1957.53	10.48	0	1000	1842.32	19.23	651	349
60	1943.88	24.71	1000	0	2043.09	9.52	0	1000	1811.52	50.7	666	334
70	1864.07	77.57	1000	0	2100.84	32.31	0	1000	1838.99	92.79	661	339
80	1939.01	33.01	1000	0	2021.53	32.99	0	1000	1907.27	69.97	686	314
90	1914.83	8.25	1000	0	2043.35	51.32	0	1000	1892.67	36.78	659	341
100	1915.04	8.23	1000	0	2086.22	28.19	0	1000	1835	30.99	668	332

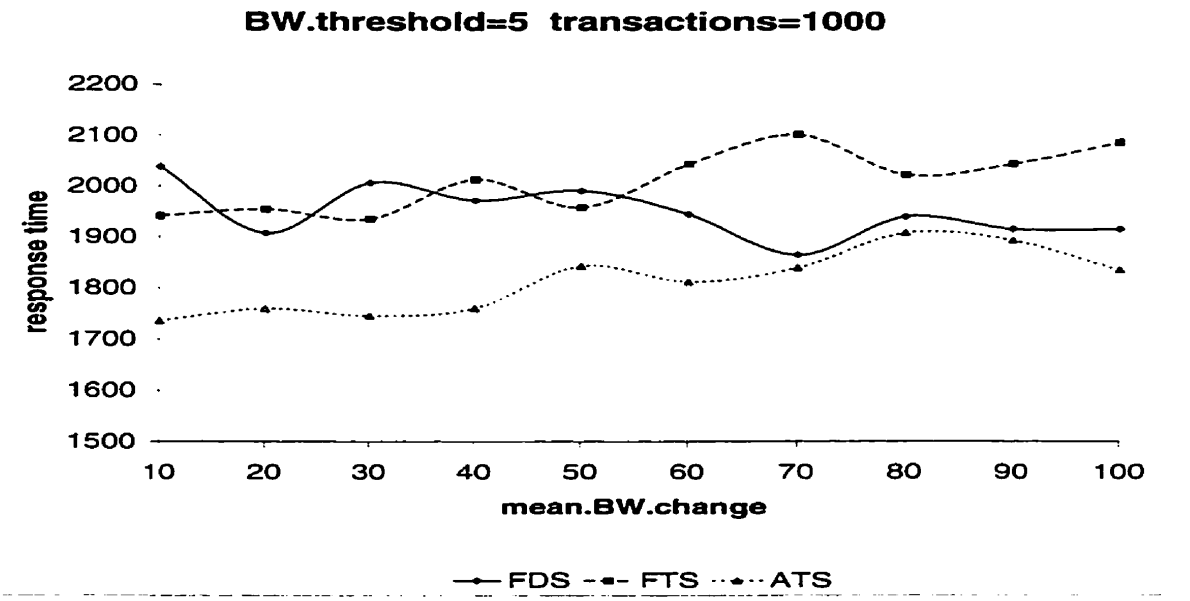


Figure 4.5: Response time vs. mean inter-event time of bandwidth change

transaction needs more data transmission than a TS transaction. Moreover, a DS transaction takes a longer time for global commitment. The ATS approach demonstrates the better outcome than the FDS approach, but worse than the FTS. The performance of ATS approach is in fact close to the FTS approach. It means that the majority of the issued transactions are scheduled as TS transactions. This can be verified from Table 4.5 that lists the data used for Figure 4.4. It should be stressed that even though the network is fully connected, the performance of the ATS may not match the one of the FTS. This is because the network connectivity, i.e. bandwidth, is not a single factor affecting the decision of the ATS. When the network is strongly connected, where a transaction is executed also depends on the volume of data transportation, which in turn relies on the cache status. For instance, if

the cache status indicates that cached data for a transaction is valid, the transaction may be also scheduled as a DS transaction, though the network is connected. Therefore, unlike the FTS approach, some transactions are generally scheduled by the ATS as DS transactions and some are TS transactions. From the figure it can also be seen that the frequency of network connection change has little impact on system performance as the network is enforced to be fully connected.

Table 4.7: The data for Figure 4.6 (BW.threshold=9 transactions=1000)

	FDS				FTS				ATS			
BW.change	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
10	6079.40	102.93	1000	0	7182.29	107.60	0	1000	6236.68	75.01	745	255
20	6287.46	3.90	1000	0	7322.37	85.68	0	1000	6299.64	77.91	774	226
30	6520.99	153.81	1000	0	7434.55	88.44	0	1000	6340.17	80.37	787	213
40	6814.15	189.09	1000	0	7616.13	92.47	0	1000	6490.33	83.85	812	188
50	6850.82	206.94	1000	0	7835.97	186.65	0	1000	6681.72	57.37	829	171
60	7168.12	71.21	1000	0	7852.64	185.83	0	1000	6610.63	119.28	828	172
70	7390.70	148.40	1000	0	7754.19	172.48	0	1000	6910.53	53.46	842	158
80	7344.33	151.48	1000	0	8012.66	138.87	0	1000	7171.02	61.53	867	133
90	7529.98	212.35	1000	0	8285.30	180.71	0	1000	7240.24	66.43	862	138
100	7770.02	216.04	1000	0	8394.13	179.57	0	1000	7425.46	249.74	860	140

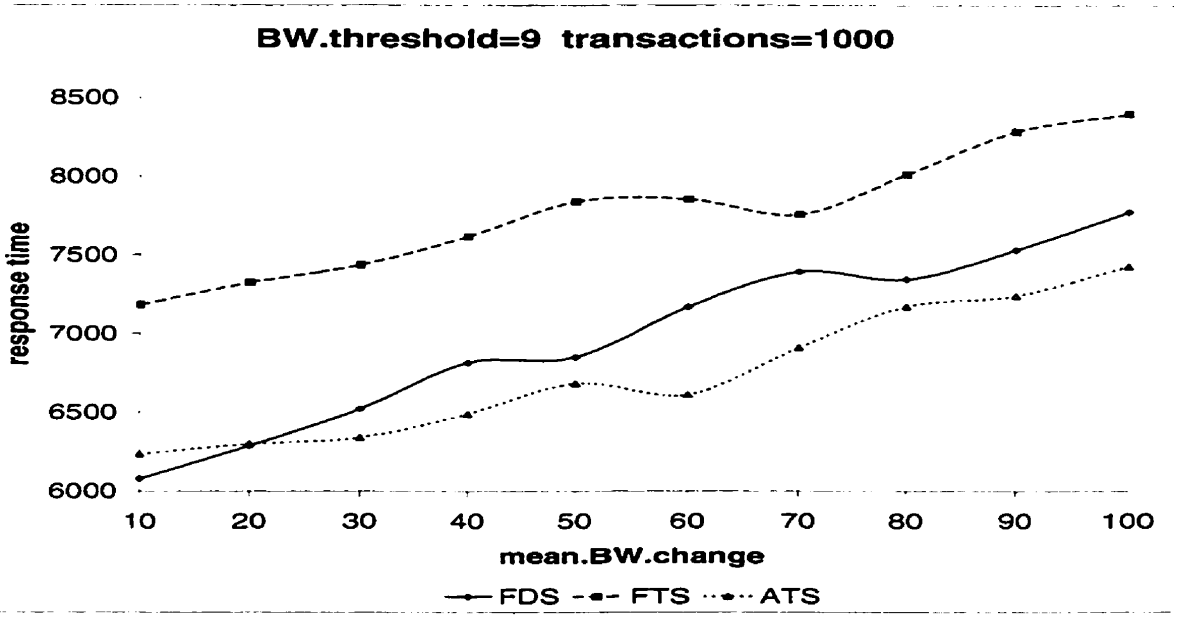


Figure 4.6: Response time vs. mean inter-event time of bandwidth change

Figure 4.5 shows the result that a mobile host has medium disconnection, i.e. disconnection ratio around 0.5. It can be seen that response time increases for all three approaches. However, because of increasing disconnection ratio, the performance of the FTS approach in general gets worse and the ATS approach is improved. The

reason is that when the network disconnection ratio increases, more transactions are scheduled as DS transactions that can be processed during disconnection. This is revealed by the data in Table 4.6. Figure 4.5 also demonstrates that the inter-event time of bandwidth change impacts the performance, but not in a consistent pattern.

The effect of network disconnection on transaction processing response time in Figure 4.6 is more obvious than Figure 4.5. In Figure 4.6, the disconnection ratio is high. Thus, with the FTS approach the chance of discarding an arrived transaction increases. On the other hand, with FDS and ATS approaches, again as cached data is used, the response time of these two approaches decrease substantially when compared to the FTS approach. From Table 4.7 it can be seen that for the ATS approach more transactions turn out as DS transactions. Figure 4.6 also shows that when the network disconnection is high, the increase of bandwidth inter-event time results in an increase in response time.

The 2nd Group Experiments and Results

Figures 4.7 to 4.9 are in the second group and show the performance (i.e. response time) of system over the network disconnection ratio. Similarly, a fixed number of transaction requests are issued. In each experiment, a mean inter-event time is given and the network disconnection likelihood is varying.

Figure 4.7 displays the system performances for the three approaches provided that changes in the network connectivity take place frequently (the inter-event time of bandwidth change is 10). It can be seen that when the network is strongly connected (BW.threshold $\leq$ 3 in Figure 4.7), the FTS and ATS demonstrate better performance than the FDS. However, with the increase of the network disconnection the response time of the FTS increases quicker than the other two approaches. It can also be seen from Figure 4.7 that the ATS approach is generally the best when the network disconnection is getting high. Nevertheless, the figure shows that the performance between the FDS and ATS has no significant difference at high network disconnection. The reason for this is largely because the inter-event time of the net-

work connection change is short. It can result in a short duration of the network connection. In the worst case, the network is partitioned for most of the time. Since the chance is very small for the ATS approach to schedule a transaction as a TS transactions, almost all the transactions are treated as the DS transactions (see Table 4.8). Therefore, the performance for the two approaches is very close when the network disconnection is high. This further suggests that when the network is nearly in a constant disconnection condition, the performance of the ATS should approach the performance of the FDS.

Table 4.8: The data for Figure 4.7 (mean BW.change=10 transactions=1000)

	FDS				FTS				ATS			
BW threshold	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
0	1528.48	26.05	1000	0	476.98	12.98	0	1000	595.01	21.37	378	622
1	1529.55	42.18	1000	0	713.80	34.72	0	1000	772.67	26.76	435	565
4	1783.53	35.86	1000	0	2074.93	62.29	0	1000	1771.80	66.44	557	443
5	1951.14	75.15	1000	0	2533.25	77.90	0	1000	2058.07	50.42	586	414
6	2160.53	59.38	1000	0	2455.43	57.73	0	1000	2080.68	35.36	681	319
7	2491.48	51.31	1000	0	2614.83	21.98	0	1000	2360.41	52.74	825	175
8	3277.61	36.68	1000	0	3844.59	106.01	0	1000	3197.72	55.92	873	127
9	7309.46	202.10	1000	0	7836.68	75.01	0	1000	7242.64	185.92	966	34

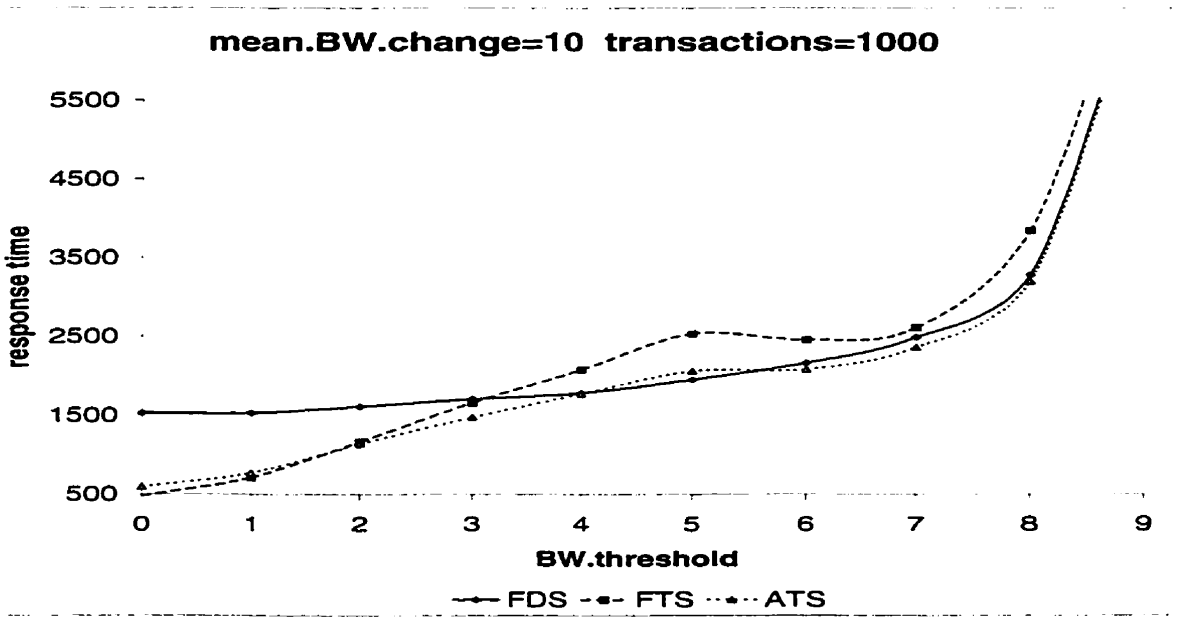


Figure 4.7: Response time vs. disconnection ratio

Figure 4.8 shows the effect of increasing inter-event time of bandwidth change on the response time of three approaches (mean.BW.change from 10 to 50). Increasing inter-event time of bandwidth change means that a state of the bandwidth remains

Table 4.9: The data for Figure 4.8 (mean BW.change=50 transactions=1000)

	FDS				FTS				ATS			
BW.threshold	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
0	1516.82	17.62	1000	0	473.55	14.96	0	1000	597.15	24.00	395	605
1	1578.43	15.19	1000	0	604.90	6.02	0	1000	755.59	3.54	413	587
2	1652.62	27.55	1000	0	900.25	42.12	0	1000	994.94	37.55	469	531
3	1784.94	37.98	1000	0	1223.22	81.76	0	1000	1334.46	24.14	563	437
4	1862.70	33.88	1000	0	1525.63	104.68	0	1000	1548.63	16.14	609	391
5	2055.81	24.74	1000	0	1936.65	50.65	0	1000	1876.11	3.08	668	332
6	2349.77	18.93	1000	0	2341.29	48.65	0	1000	2149.22	32.56	715	285
7	2856.24	31.90	1000	0	2885.53	52.38	0	1000	2666.02	12.54	774	226
8	3720.94	21.94	1000	0	3802.89	26.46	0	1000	3570.79	54.42	811	189
9	8410.19	354.66	1000	0	8788.98	80.05	0	1000	7736.03	307.66	873	127

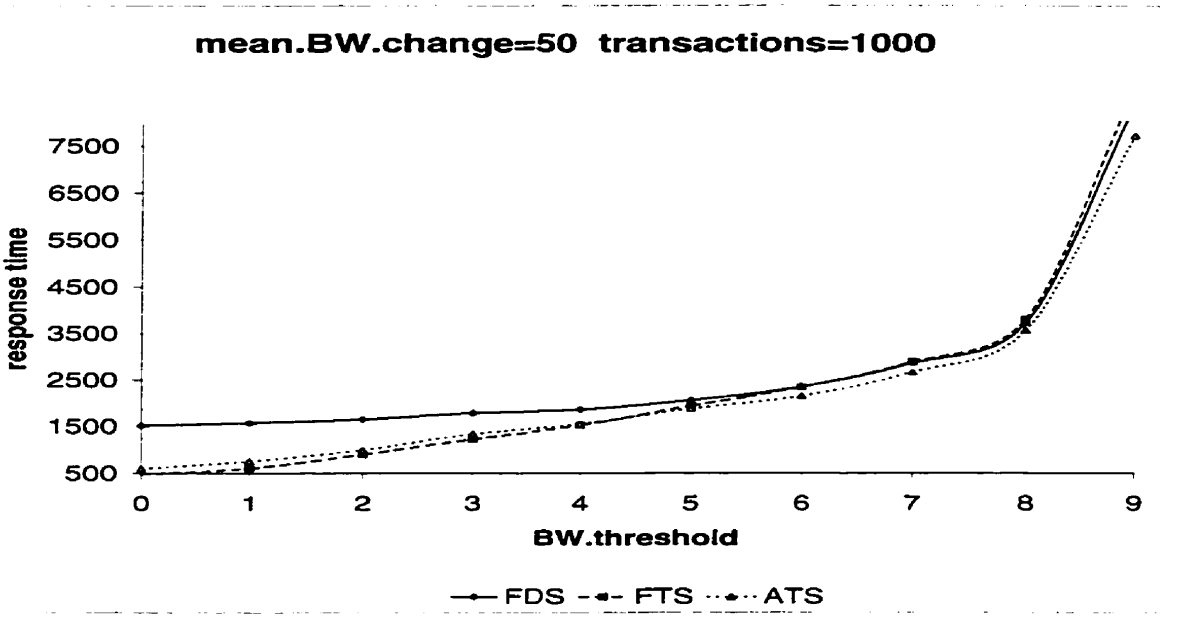


Figure 4.8: Response time vs. disconnection ratio

for a longer time before changing to another state. As we can see, similarly, when the network connectivity is strong, the FTS demonstrates the best performance, whereas the FDS demonstrates the worst. However, in this experiment, strongly-connected network condition shifts up to BW.threshold=5 compared to BW.threshold=3 in Figure 4.7. This illustrates that the communication bandwidth remains at a higher level before that point. This result somehow depends on the pattern of bandwidth change. In this simulation we assume that it has a uniform distribution. If the distribution differs, the outcome may differ. An extreme case can help to understand this. Suppose the bandwidth happens to change to a high level and such changes will never take place again, then the network remains well connected. Therefore, the bandwidth threshold has little impact on the system per-

formance. On the other hand, if the bandwidth remains at a lower lever, the network has a longer partition time.

Due to a longer inter-event time of bandwidth change, the duration that the network remains in connection increases once the bandwidth changes into a high level. Thus, even if the disconnection ratio is high, there still exists some chance for a transaction to be scheduled as a TS transaction. This is unlike the FDS as it does not rely on the processing of a fixed host even though the network is strongly connected. That explains that the performance of the ATS approach has a bigger difference than the one of the FDS at a high network disconnection status.

Table 4.10: The data for Figure 4.9 (mean BW.change=100 transactions=1000)

	FDS				FTS				ATS			
BW.threshold	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#	mean RT	std.dev	DS#	TS#
0	1512.41	16.13	1000	0	476.51	10.84	0	1000	536.53	24.43	399	601
1	1561.99	20.80	1000	0	612.98	25.77	0	1000	728.91	15.72	419	581
2	1597.31	37.97	1000	0	941.86	51.04	0	1000	1007.60	14.92	468	532
3	1723.15	27.33	1000	0	1249.62	54.21	0	1000	1317.44	55.36	563	437
4	1821.43	42.75	1000	0	1636.99	66.95	0	1000	1540.86	34.01	598	402
5	1979.97	35.42	1000	0	1971.61	81.27	0	1000	1855.03	1.77	673	327
6	2237.43	23.19	1000	0	2372.81	44.78	0	1000	2063.07	10.67	680	320
7	2648.84	45.50	1000	0	2883.27	40.16	0	1000	2518.85	36.59	772	228
8	3555.59	56.51	1000	0	3646.26	86.73	0	1000	3256.18	39.52	817	183
9	6624.46	34.39	1000	0	7681.72	57.37	0	1000	5851.08	74.27	833	167

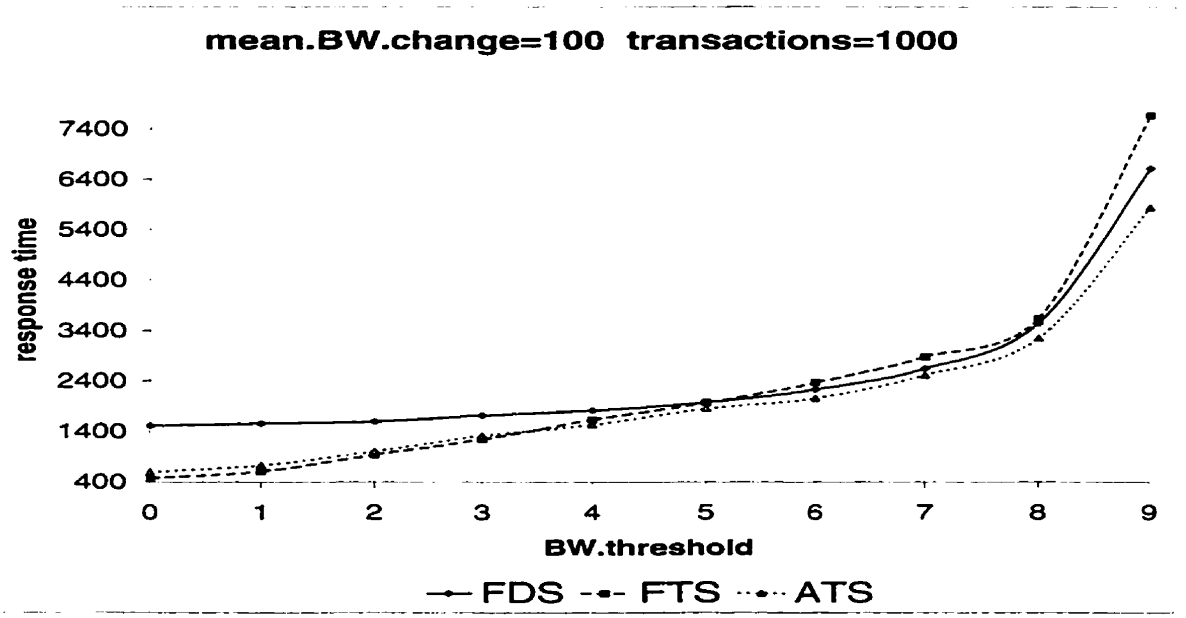


Figure 4.9: Response time vs. disconnection ratio

The explanation is further confirmed by Figure 4.9. For the same reason, the ATS

demonstrates a more significant performance difference from the FDS approach when the disconnection ratio is high. In short, from the figures in this group, it can be seen that the three approaches roughly demonstrate identical performance patterns. That is, response time increases with the increase of the network disconnection ratio. Moreover, when the network disconnection chance is lower, the FTS performance is superior to the other two approaches. When the network disconnection chance is higher, the FTS performance deteriorates. These results are consistent with the ones produced in the first group.

The 3rd Group Experiments and Results

Table 4.11: The data for Figure 4.10 (BW.threshold=0 transactions=1000)

	FDS				FTS				ATS			
BW.change	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#
10	288.22	9.36	1000	0	29.02	1.13	0	1000	258.55	19.14	394	606
20	292.76	4.15	1000	0	28.90	0.52	0	1000	252.70	10.20	383	617
30	281.98	2.92	1000	0	28.75	1.47	0	1000	249.13	7.51	353	647
40	288.97	6.02	1000	0	29.06	0.98	0	1000	254.10	2.20	398	602
50	288.36	1.78	1000	0	28.86	0.80	0	1000	251.09	7.22	394	606
60	286.65	6.59	1000	0	29.07	0.63	0	1000	264.08	5.70	412	588
70	288.08	8.43	1000	0	29.81	0.81	0	1000	262.17	10.06	405	595
80	289.53	3.06	1000	0	29.29	0.63	0	1000	254.40	8.82	388	612
90	288.61	3.50	1000	0	28.35	0.15	0	1000	251.71	9.27	366	634
100	286.07	5.26	1000	0	28.57	0.49	0	1000	257.72	11.08	366	634

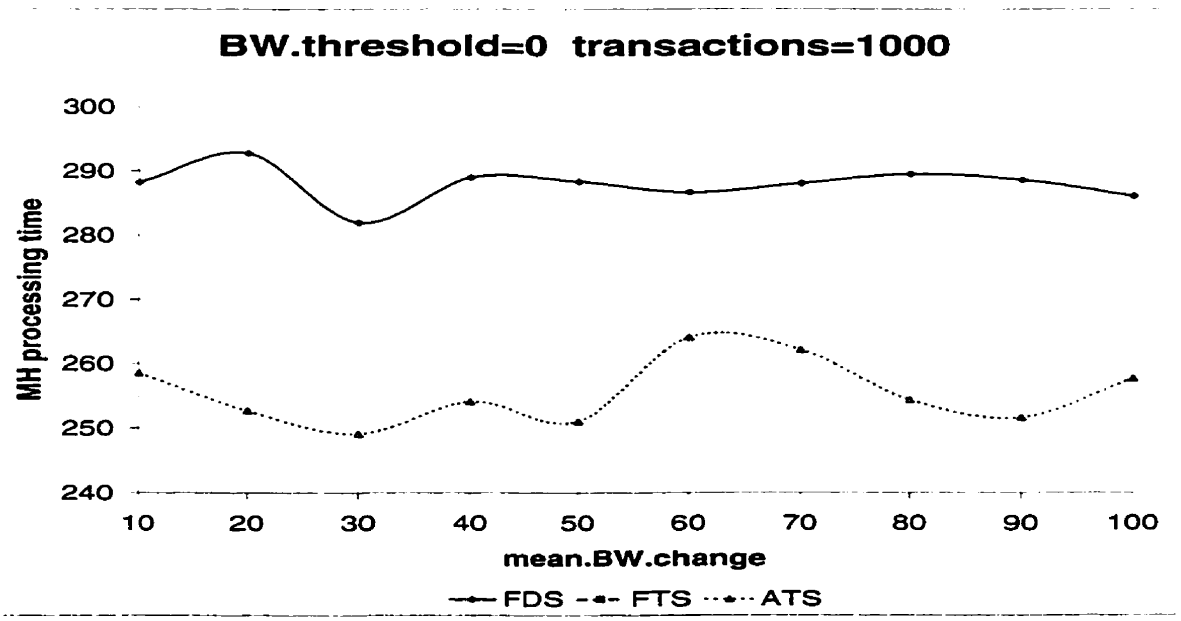


Figure 4.10: Processing time vs. mean inter-event time of bandwidth change

The battery energy consumption is a big concern in designing mobile transaction schemes. Generally, the battery consumption is a direct proportion to the processing time spent on a mobile host. Experiments in the third group are conducted to compare this time with different approaches. Figures 4.10 to 4.12 show the simulation results for a given network disconnection ratio i.e. BW threshold=0, 5, and 9, respectively.

Table 4.12: The data for Figure 4.11 (BW.threshold=5 transactions=1000)

	FDS				FTS				ATS			
BW change	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#
10	290.09	1.94	1000	0	26.00	1.35	0	1000	279.33	6.15	611	389
20	286.10	4.72	1000	0	25.96	0.25	0	1000	279.54	3.64	652	348
30	287.14	2.23	1000	0	25.96	1.49	0	1000	280.74	2.28	677	323
40	289.57	3.52	1000	0	26.62	1.28	0	1000	283.72	4.32	641	359
50	286.19	0.73	1000	0	26.32	1.01	0	1000	281.99	5.88	651	349
60	291.20	4.80	1000	0	25.48	0.93	0	1000	282.27	3.60	666	334
70	296.11	3.56	1000	0	26.55	0.56	0	1000	287.65	7.05	661	339
80	289.53	3.30	1000	0	26.54	0.36	0	1000	285.35	2.73	643	357
90	287.66	4.01	1000	0	26.01	1.11	0	1000	282.09	5.37	659	341
100	289.07	5.10	1000	0	25.28	0.70	0	1000	281.39	6.20	668	332

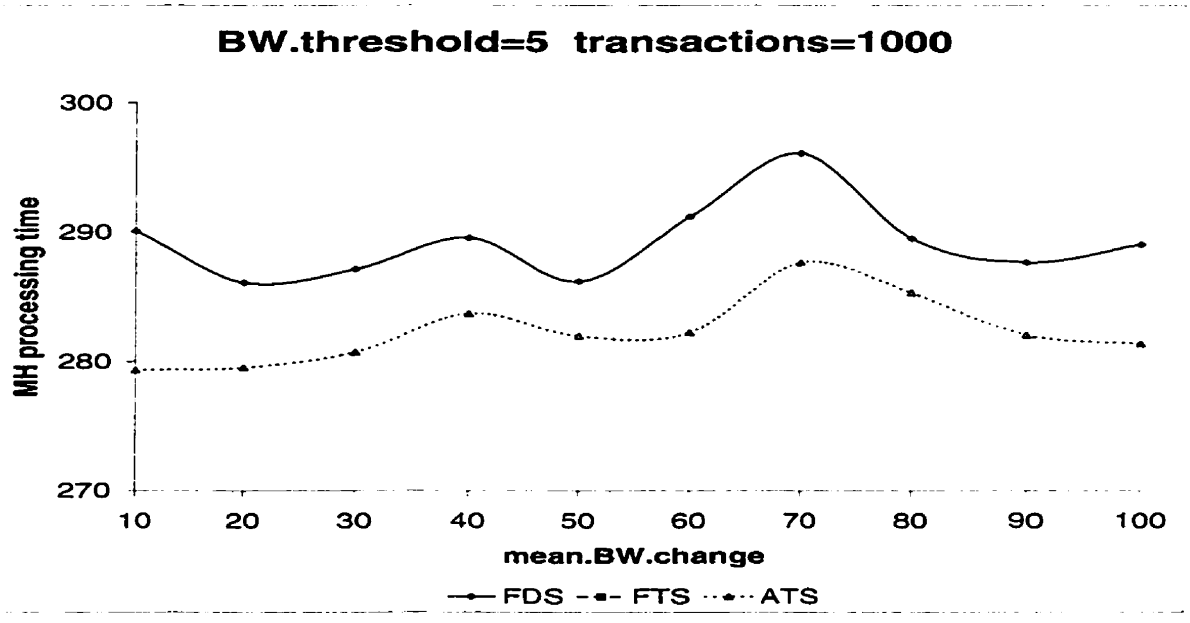


Figure 4.11: processing time vs. mean inter-event time of bandwidth change

Among the three approaches, the FTS takes significantly shorter time as it performs all processing on a fixed host. As a result, the figures only show results for the FDS and ATS approaches. It can be seen from Figure 4.10 to 4.12 that the FDS approach generally takes more processing time on the mobile host side than ATS.

Figure 4.10 shows that when the network is strongly connected, the FDS approach takes more processing time on a mobile host than the ATS approach. This is because in this condition, with the ATS approach, some transactions are scheduled as TS transactions so that the time spent on a mobile host is reduced. It can be seen from the figure that the inter-event time of bandwidth change has certain effect on the processing time, but this simulation experiment does not produce an obvious pattern.

Table 4.13: The data for Figure 4.12 (BW.threshold=9 transactions=1000)

	FDS				FTS				ATS			
BW.change	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#	mean PT	std.dev	DS#	TS#
10	289.69	2.07	1000	0	54.04	1.98	0	1000	281.38	2.36	745	255
20	288.82	1.74	1000	0	53.42	1.21	0	1000	282.37	1.29	774	226
30	289.75	1.19	1000	0	51.49	0.81	0	1000	285.53	2.36	787	213
40	289.26	1.06	1000	0	54.81	1.13	0	1000	286.81	1.83	812	188
50	289.06	0.92	1000	0	54.16	0.15	0	1000	285.03	1.01	829	171
60	289.00	0.70	1000	0	53.66	1.37	0	1000	286.23	0.14	828	172
70	289.24	1.19	1000	0	54.86	0.52	0	1000	286.48	2.23	842	158
80	288.97	1.19	1000	0	56.08	2.36	0	1000	287.97	2.24	867	133
90	289.79	1.65	1000	0	53.84	1.02	0	1000	288.98	1.37	862	138
100	289.56	0.67	1000	0	55.01	0.34	0	1000	289.28	0.72	860	140

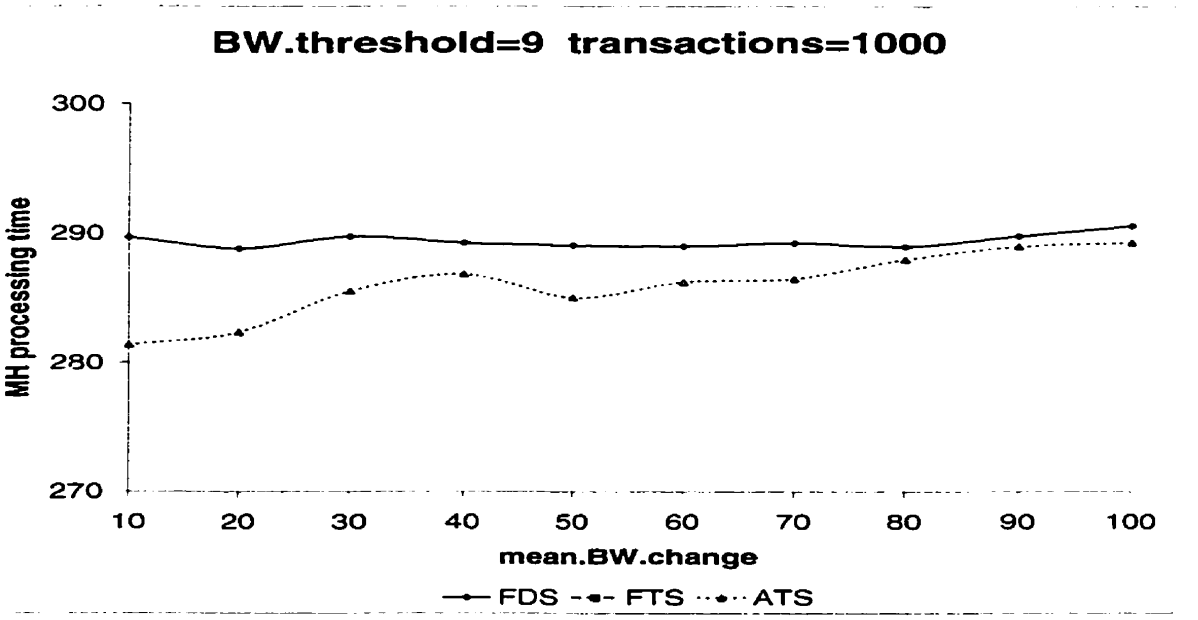


Figure 4.12: Processing time vs. mean inter-event time of bandwidth change

When the network disconnection occurs (see Figure 4.11), the ATS approach still takes less transaction processing time on a mobile host than the FDS. However, the time for the two approaches converge as mean inter-event time of bandwidth change increases. As we can see from Table 4.12, with the ATS approach more transactions

are scheduled as DS transactions. Hence, it is natural for the ATS to have an increase in processing time. This can be more clearly viewed from Figure 4.12.

Table 4.14: The data for Figure 4.13 (mean BW.change=30 BW.threshold=9)

	FDS			FTS			ATS		
simulation time	MT#	DS#	TS#	MT#	DS#	TS#	MT#	DS#	TS#
100	69	69	0	30	0	30	88	66	22
200	138	138	0	58	0	58	174	129	45
300	206	206	0	84	0	84	261	197	64
400	281	281	0	121	0	121	348	262	86
500	345	345	0	143	0	143	439	333	106
600	407	407	0	163	0	163	530	401	129
700	475	475	0	197	0	197	606	451	155
800	554	554	0	225	0	225	705	524	181
900	616	616	0	242	0	242	787	594	193
1000	688	688	0	275	0	275	880	664	216

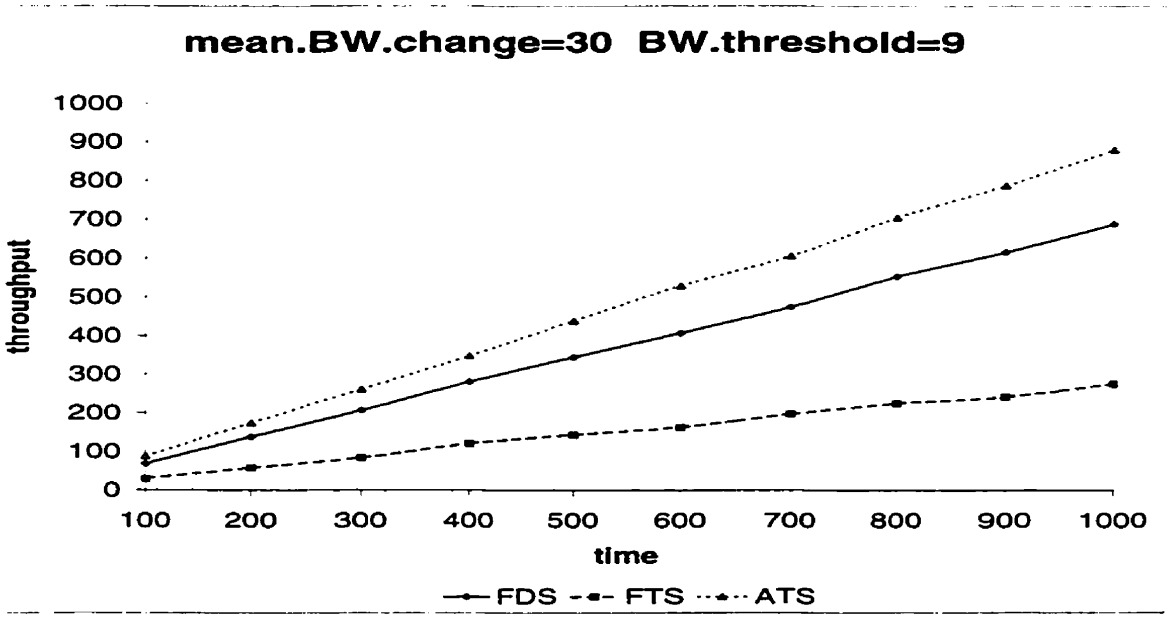


Figure 4.13: System throughput vs. simulation time

Figure 4.12 gives the outcome when the network disconnection is high. With the ATS approach, a high disconnection results in more transactions that are scheduled as DS transactions (see Table 4.13). This therefore raises the transaction processing

time on a mobile host. It can be stated that when the network disconnection ratio is very high, with the ATS approach the processing time on a mobile host is approaching the processing time of the FDS approach as few transactions are scheduled as TS transactions.

The 4th Group Experiments and Results

Table 4.15: The data for Figure 4.14 (BW.threshold=0 simulation time=1000)

	FDS			FTS			ATS		
BW.change	MT#	DS#	TS#	MT#	DS#	TS#	MT#	DS#	TS#
10	663	663	0	1190	0	1190	1049	346	703
20	636	636	0	1182	0	1182	1072	358	714
30	606	606	0	1167	0	1167	1049	343	706
40	669	669	0	1143	0	1143	1090	396	694
50	668	668	0	1167	0	1167	1106	372	734
60	638	638	0	1158	0	1158	1054	341	713
70	640	640	0	1224	0	1224	1059	319	740
80	690	690	0	1152	0	1152	1020	290	730
90	667	667	0	1178	0	1178	1046	350	696
100	633	633	0	1190	0	1190	1070	341	729

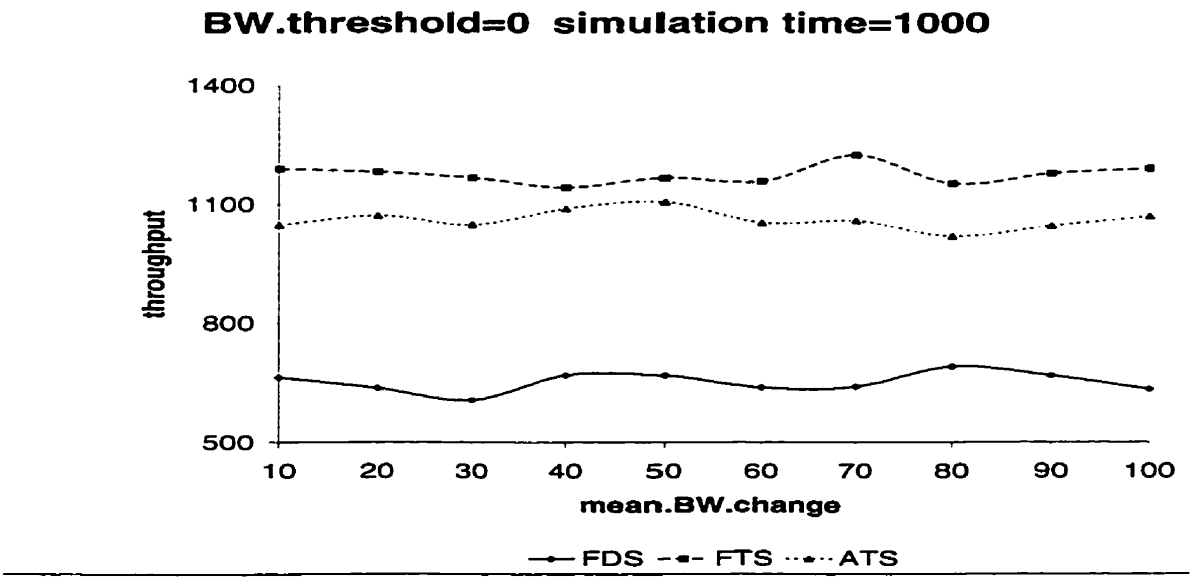


Figure 4.14: System throughput vs. mean inter-event time of bandwidth change

The final group includes Figures 4.13 to 4.15. They show the experiment results for system throughputs in terms of the number of completed transactions. The first experiment investigates the throughput versus simulation time in the condition that both the inter-event time of bandwidth change and the network disconnection ratio are fixed.

Table 4.16: The data for Figure 4.15 (BW.threshold=5 simulation time=1000)

	FDS			FTS			ATS		
BW.change	MT#	DS#	TS#	MT#	DS#	TS#	MT#	DS#	TS#
10	505	505	0	599	0	599	656	425	231
20	520	520	0	573	0	573	690	426	264
30	474	474	0	560	0	560	633	401	232
40	477	477	0	538	0	538	640	425	215
50	508	508	0	577	0	577	634	447	187
60	556	556	0	572	0	572	639	426	213
70	491	491	0	568	0	568	645	436	209
80	123	123	0	67	0	67	148	122	26
90	119	119	0	65	0	65	149	119	30
100	126	126	0	67	0	67	130	116	14

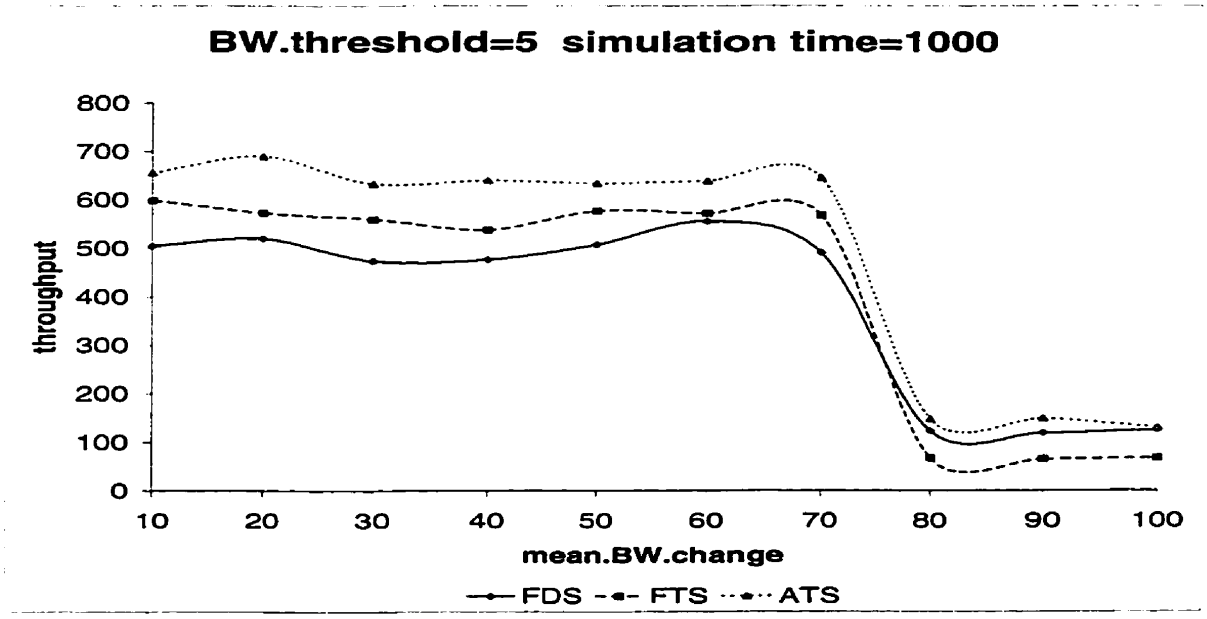


Figure 4.15: System throughput vs. mean inter-event time of bandwidth change

Figure 4.13 presents the result when the disconnection ratio is high (BW.threshold=9).

The purpose of this experiment is twofold. The first one is to evaluate the correctness of the simulation model. Theoritically, given a transaction processing policy, its performance in terms of throughput should be opposite to its response time with the same condition. This can be verified compared to the result shown in Figure 4.6. The second purpose of this experiment shows that when the network connection is weak, with the increase of simulation time, the ATS approach consistently demonstrates a better performance than other two approaches.

However, as shown in Figure 4.15, when disconnection exists, the ATS can produce better throughput over the other two approaches because the FTS is unable to accept more transaction requests during network disconnection. Although the FDS can carry on transaction processing with cached data, it ignores the fact that the network may get connected occasionally. Figure 4.15 shows a sharp performance drop after the inter-event time of bandwidth changes from 70 and on. This implies that the network has a high disconnection. This can happen when the initial bandwidth level is low and then remains there for a long time. In Figure 4.15, this means that when the BW.change is greater than 70, the bandwidth happens to be initially lower than 5 and then without much changes during the pre-defined simulation time interval. This again has something to do with the distribution of bandwidth change (or a longer simulation time should be applied), and is left for future investigation.

Figure 4.14 and 4.15 show the throughputs of the system for given simulation time interval (close.time=1000), but with different network disconnection ratios, respectively. Figure 4.14 illustrates that without disconnection, the FTS approach has the highest throughput, while the FDS the lowest and the ATS approach is in between the two. This result is consistent with the result shown in Figure 4.4 as an approach with a lower throughput can have a longer response time.

To summarise, the simulation experiments performed in this research investigate the performance of three transaction processing policies under the assumption that network disconnection occurs. The experiments are conducted from several perspectives by adjusting model parameters, such as, average inter-event time of bandwidth

change and disconnection ratio. The simulation results show that if there is little or no network disconnection, the FTS has the shortest response time and the highest system throughput. This is not surprising because the system operates at or near the traditional fixed network environment. However, as network connectivity deteriorates, the FDS produces better system throughput and response time than the FTS. In general, the ATS approach proposed in this research produces better performance in terms of overall response time, elapsed processing time of a mobile host, and total number of transactions completed by the system.

Chapter 5

RELATED WORK

5.1 Adaptive Approaches in Mobile Computing

As mentioned in Chapter 1, an adaptive approach is a strategy that allows a computing system to perform certain actions based on its perception of the current system environment. Researchers working with this type of approach have a common belief that if protocols and applications are designed to be aware of changes in the environment, such as network connection quality and mobile patterns, they may be able to adapt their performance to the new conditions. Nevertheless, under the heading of adaptive approach, the research objectives and target application domains are diverse. In general, the research interests in current projects cover three levels in terms of a computing system hierarchy which are: (i) network message transportation level, (ii) operating system level, and (iii) application level.

5.1.1 Network Message Routing Level

With the introduction of wireless communication and mobile units, as it pointed out in [Ach95], a pre-defined hierarchical addressing scheme in existing internetworking protocols cannot support mobile computing efficiently

An initial response is to extend existing protocols so that mobility can be taken into account and backward compatibility with the existing network infrastructure is maintained.

Message routing under the mobile internetworking protocol called Mobile IP [JM96] uses an "IP within IP" mechanism. With this mechanism [RB95], each mobile host can have different Internet addresses that are used when the mobile host changes locations. There is a single address, called the *home address*, that is permanent and designated by its home network router, known as the *home agent*. When a mobile host moves to a new network, it must apply for a temporary address, called the *foreign address*, from a local network router, known as the *foreign agent*.

To support transparent movement of mobile units throughout the Internet, messages to a mobile unit are always routed to its home agent. This is referred to as the *basic triangle routing method*. Upon arrival of a message at a home agent, the home agent adds the foreign address of a mobile unit to its home address package so that message sent to the mobile unit can be forwarded to its current location.

The basic triangle routing method can be considered as a static routing method because messages are always routed through the home agent of a mobile unit. This method uses a fixed routing strategy and does not exploit current environment information. For instance, suppose two mobile units are far away from their home locations, even though they are physically close to each other, the messages between them have to be sent back to their home agents. The research in [JM96, Yua93] made efforts to develop adaptive mobile IP communication protocols. The adaptive mobile IP communication protocols focus on the dynamic optimisation of message routing based on historical routing information.

The approach in [JM96] explores the reuse of the routing information stored in the caches of routers. If route information previously used for a mobile unit is still available, messages to a mobile unit will not go through its home agent, instead a short-cut may be applied.

The packet routing scheme in [Yua93] is concerned about efficient propagation of routing information. The network is distinguished into two parts: namely, friend network and non-friend network. A friend network is the one that has substantial traffic to and from the mobile unit. The routing information is only propagated to

friend networks. Therefore, traffic from a friend network can achieve optimal routing, while others need to be forwarded to the mobile unit by its home agent. In this way, the network bandwidth used for the routing information update is expected to be greatly reduced. Nevertheless, the determination of friend networks is a key point in the design and success of the scheme. It relies on dynamic traffic pattern analysis. For each traffic pattern, there will be an optimal routing scheme that adapts to the corresponding traffic pattern.

In [RB95] a dynamic routing method, called *adaptive enhanced triangle routing*, is proposed. Unlike the approach in [Yua93] that separates network into two parts, this routing method dynamically breaks mobile units into two groups based on a piece of mobility information, called the *calling/mobility ratio*.

The mobile units in a group are considered as active partners of a given mobile unit if their calling/mobility ratios are higher than a pre-defined threshold. The separation of mobile units makes the optimisation of overall message routing performance possible. The idea is that each time when a mobile unit changes its location, its partners are informed. In this way, messages from a partner of a mobile unit can directly be routed to the mobile unit. For a non-partner it must resort to the basic triangle routing method, which travels via the home agent of a mobile unit and takes more time.

Adaptive communication protocols are also investigated in the WAMIS project [ABB+96]. This project focuses on *ad hoc* mobile environments where all the units in the system are mobile. The target application is multimedia information services. Because there is no fixed network, message routing links must be dynamically established. The applications running under such environments where the quality of network services is constantly fluctuating, must be adaptive. In this project, a group of wireless sub-network control algorithms are implemented. The application algorithms for voice and video source coding are rate-adaptive, i.e. they select proper compression rates based on currently available bandwidth captured by a underlying networking module.

5.1.2 Operating System(OS) Level

Some researchers believe that adaptation involves the co-operation between the OS and applications. Even though the experimental platforms are different, the ideas behind the research in [DBCF94, NPS95, MES95] are similar to each other. That is, the OS provides mobile users with a set of Application Program Interfaces (APIs) and takes the responsibility for monitoring the availability of system resources upon a user's request and notifying the user if a change takes place.

To implement this approach, functionality of operating systems must be extended. On the other hand, an application must determine required system resources and define the tolerable ranges of resource variation in advance.

The authors in [DBCF94] develop a set of control API based on a specific network. The environment factor monitored by the server is the bandwidth of the network. The authors investigate the effect of a varying environment on the performance of image retrieval and demonstrate the feasibility of their adaptive method.

In [NPS95], the factors considered in the system environment are more general than in [DBCF94]. A system environment is modelled as a set of resources available for applications. Each resource is measured by a scalar value. A user can specify interested resources and the tolerance range through a set of APIs. A server informs the user if the availability of the required resources are out of range. Upon receiving notification, the application may change its performance level by invoking pre-defined procedures.

The work reported in [MES95] is based on the Coda file system. To support varying degrees of network connectivity in a mobile computing environment, many system aspects, including communication, cache validation, update propagation and cache miss handling are modified.

[SAW94] reports a scheme supporting adaptive applications that are aware of their environment context such as locations, available resources and network connectivity. The approach assumes that a mobile unit can periodically send an identifying packet that permits accurate location monitoring. The system captures mobility information

about physical movement of a mobile unit, which serves as a "context reminder". This notifies applications of location changes and in turn triggers context-dependent actions. The pre-defined actions are represented like IF-THEN rules that specify how an application should be reacted to within a new context.

5.1.3 Application Level

Some methods can be considered as high-layer, or application-level adaptation, because they concentrate on proper design of applications by making use of dynamic information rather than modifying the underlying system functions. The research reported in [VB94, AIB94] are examples of application level adaptation. The application domain is Web document retrieval. The concepts about Web documents and Uniform Resource Locators (URLs) are extended. Information is first organised as a set of pages, which includes location-dependent control information. A URL is then treated as a kind of location variable. It is instantiated each time when a client request is issued at a particular location. A Web server checks the current value of a location variable and delivers only those documents relevant to the user's location.

Moreover, [Kat94, IV94] describe adaptive concepts and methods from the viewpoint of developing wireless information systems. Katz [Kat94] stresses the importance of adaptability in mobile computing. A number of common uses for adaptability, including adaptive power control, adaptive channel allocation, adaptable network topology and adaptive bandwidth, are addressed. The author claims that mobility requires adaptability and describes the infrastructure of a wireless information systems, an implementation methodology, and the engineering technology to support its operation.

The system reported in [IV94] is more specific. It assumes that a mobile unit can work in active mode and doze mode. Doze mode is a low energy consumption mode that still allows a mobile computer to listen to broadcast messages on the network. Information services can be provided in publishing mode and on-demand mode. If certain information is becoming "hot", it can be published from a wireless channel;

otherwise it will be provided upon request. The fundamental technique of adaptive access protocols is to schedule information services dynamically based on statistics gathered from user service requests.

To summarise, we can see that an adaptive approach has attracted the attention of many researchers in the mobile computing area. The approach can be applied to tackle the issues of mobile computing in many respects. In general, three major tasks have to be performed in developing an adaptive application. The first one is to collect environment information. Environment information can be captured in different ways, at different system levels, and for different purposes. As information is captured at lower levels (e.g. OS, networks) a wider range of applications can be supported. In the long run, as these layers are redeveloped to better support mobile environments, the ability to capture mobile information can be taken into account as part of the design objectives. The second task is to identify potential actions for different conditions. Adaptation can only occur if some options exist. The third task is to develop a decision-making model. Potential actions can be assessed based on the model and available information so that an optimal action can be selected. The research presented in this thesis follows this methodology.

5.2 Mobile Transaction Processing

In general, research on mobile computing is still in its early stages. As a result, a common model for mobile distributed transaction processing has not yet evolved. Current research in this area is mostly focused on unit disconnection and data consistency issues. The major approaches dealing with the issues of transaction processing in mobile environments are reviewed below.

5.2.1 Yeo & Zaslavsky Approach (YZA)

Yeo and Zaslavsky [YZ94a, YZ94b] propose a mobile transaction processing scheme from the perspective of distributed multidatabases. The scheme assumes that a

shared global database is distributed on stationary machines. A mobile transaction is considered as a transaction that only accesses the global database.

A mobile transaction initiated by a mobile unit is always submitted to a stationary machine with required control information. The stationary machine serves as an agent (co-ordinator) on behalf of the mobile unit to schedule and execute the transactions. The result or an acknowledgement message will be returned to the mobile unit after the transaction is committed or aborted.

This approach considers mobile transaction processing as a remote procedure call (RPC). The advantage is that once a transaction is submitted, a mobile unit does not have to wait for the completion of the transaction before it moves away from its current location or becomes disconnected. However, this approach is not appropriate to a more general mobile database environment where part of the data is on the mobile unit and part of the data is on the fixed hosts. In such an environment, because of autonomous operations on the mobile unit during disconnection, maintaining data consistency and system performance becomes crucial. With the YZA approach, however, this issue is not addressed.

5.2.2 Pitoura & Bhargava Approach (PBA)

The work reported in [PB94, PB95] allows transactions to operate on both local data stored at the user's computer and remote data. The approach largely investigates the issue of data consistency in a mobile environment. To handle data consistency, a database is dynamically grouped into clusters. Each cluster is a unit maintaining full data consistency. In this way, data consistency can be distinguished from different levels.

Fundamentally, data consistency can be measured in two aspects: *intra-cluster consistency* and *inter-cluster consistency*. The data in a single cluster must be consistent at any time; while replicated data of inter-clusters may be inconsistent at some time. Therefore, the data in a cluster is *strict consistent*; whereas the data between clusters may be *weak consistent*.

Accordingly, the transactions are of two types: *weak* and *strict*. A weak transaction only accesses data within a cluster, while a strict one accesses globally consistent data. Weak transactions can be performed locally and have two commit points, a local commit when network communication is poor and a global commit after cluster merging.

To control concurrent access, a pessimistic algorithm is applied to a intra-cluster scope because a weak-write that updates the data in a cluster is assumed to be performed in a normal network condition; whereas an optimistic algorithm is used for a inter-cluster scope, i.e., the validation of the updates is delayed until clusters are connected and merged. Therefore, from the perspective of traditional distributed database management, the concurrency control mechanism used in this work can be viewed as an integration of pessimistic and optimistic algorithms.

5.2.3 Narasayya Approach (NA)

Narasayya [Nar94] points out that a mobile unit can play three roles: a dumb terminal, a full-fledged station, or a pre-processing front-end. He believes that the placement of data in a mobile distributed environment is important because wireless communication costs becomes a dominant factor in the overall performance of mobile transaction processing. After analysing two extreme situations, that is, a mobile unit as a dumb terminal or a full-fledged workstation, the author claims that both of them can suffer from the penalty caused by low bandwidth wireless communication. From this perspective, an approach called batched transaction method(BTM) is proposed.

The BTM method assumes that a mobile unit has computing resources that are in between a dumb terminal and a full-fledged workstation, and therefore plays the role of a pre-processor. Rather than executing a mobile transaction entirely on a mobile unit or on a fixed machine, as a compromise, a mobile unit takes part of the responsibility. With the BTM method, the read/write operations of a mobile transaction are executed on a mobile unit using cached copies of the data, whereas the commit operation is performed by a fixed host. When a transaction reaches its

commit point, the "entire transaction" is sent to a fixed database server and the server attempts to commit the transaction.

As we can see, although the BTM uses cached data for transaction processing, it does not allow local commitment. Only one-level commitment, i.e. global commitment is applied. Moreover, the method requires an optimistic concurrency control algorithm to maintain database consistency. Optimistic concurrency control favours frequent disconnection and reduces the number of messages sent between the mobile unit and the fixed network. However, the approach can benefit only when the read/write ratio of database operations is high. Otherwise, restarting transactions due to a high probability of using stale data will offset the benefit. In the worst case, transaction commitment may never succeed. The author notes that there is no upper bound for a successful commit.

5.2.4 Elmagarmid, Jing & Bukhres Approach (EJBA)

The work reported in [EJF94, EJB95, JBE95] assumes that a transaction can be submitted interactively to multiple servers rather than a single server. Under this assumption, the authors investigate how to use replicated data to reduce the number of messages among multiple servers while a mobile unit is moving.

A transaction can involve multiple co-ordinators during its lifetime while it is moving. As a result, read locks can be granted from different data server sites. Thus, additional messages have to be sent to those sites in the transaction commit phase to release the locks. To reduce locking messages for the global commit of a mobile transaction, the traditional optimistic Two Phase Locking (2PL) algorithm [CL91] is extended for replicated mobile database environments.

Fundamentally, the goal is to minimise the numbers of locking messages among servers by exploiting the commutative semantics of operations in particular application domains, such as the problem of resource allocation and inventory applications. To reduce messages, the algorithm requires the transmission of log information from a mobile unit to its current co-ordinator through a wireless channel when the unit

decides to commit a transaction. Moreover, the replicated database must meet specified commutative properties. That is, the operations performed at one site can be carried out at another site without the violation of overall data integrity. A typical example of resource allocation problems and relevant commutative operations can be found in [EJF94].

5.2.5 Walborn & Chrysanthis Approach (WCA)

Like the NA approach, the WCA approach requires the data items be downloaded from a global database and cached locally by a mobile unit. To support autonomous processing and commit on the mobile units despite temporary disconnection, the authors in [Chr93, WC95] investigate the issues of concurrency and cache coherency problems in a mobile environment.

When a mobile unit accesses an item, a cache request is sent to the database server by explicitly specifying a selection criteria and consistency conditions. The selection criteria indicates the object to be cached and the size of the cached data, while consistency conditions are similar to traditional integrity constraints that need to be satisfied to maintain the consistency of the entire object.

The objective is to make transaction processing independent of the rest of the system as much as possible so that computations can be carried out on a mobile unit during a period of disconnection. This is achieved by making use of semantic information of transaction applications and projecting the data items necessary for the operations in a transaction. The method requires a unit as a client to take the responsibility for specifying a cached object and its size.

The fundamental idea of this approach is to narrow the size of cached data items by splitting large and complex data into small pieces; treating cached data items as the unit of reconciliation of updates; and supporting unilateral commitment of a transaction executing on a mobile unit. The approach states that a cached data object is logically removed from the database and, therefore, no longer accessible at the server until it is released. Thus, the maintenance of data consistency can be

equated to a a traditional locking mechanism.

5.2.6 Ganguly & Alonso Approach (GAA)

Ganguly and Alonso [GA93] deal with read-only transaction (i.e., query). Aiming at query optimisation, the approach introduces a new optimisation criterion to minimise the resource utilisation on the server as well as the energy consumption on a mobile client.

A number of transaction strategies, or query plans, are generated at compile-time. The best plan is selected based on the current client/server system condition. A query can be processed either on the server side or client side based on the server's workload and the level of energy on the mobile client. The approach attempts to make an adaptation to varying connectivity conditions and exploits the resources on both the server and client sides. Concurrency control issues in this approach are not addressed because the transactions are read-only.

5.2.7 Coda Approach (CA)

Coda [MES95] is a distributed file system with the intention to support mobile units across varying network connectivity. The use of a client cache is a major component of the approach. In normal conditions, a working set of files is prepared for autonomous processing during disconnection. A user always uses a local copy as long as the cached data are validated. Although it does not deal with database issues directly, the underlying techniques, especially the cache management scheme, can provide significant contributions to mobile transaction processing.

A caching scheme has to deal with three common issues: *modification propagation*, *cache validation* and *cache misses*. Coda deals with modification propagation by so-called *trickle reintegration*, which is an asynchronous write-back mechanism. The write-back of modified data on a client cache can defer for many minutes or hours. To deal with stale data, Coda uses two methods. If the network connection is normal, a server call-back mechanism is used. That is, if a primary copy on the server has

been changed, the server sends a message that cached data is no longer valid to the clients. During a disconnection period, cached data is used. When switching from disconnection to reconnection, a client initiated approach is used, that is, the mobile host sends a query to invalidate all cached items before using them.

Coda uses a hierarchical method to invalidate cached items. The invalidation is performed from top to bottom. If the data at a upper level has not been modified, it is not necessary to go through the rest of the levels. In this way, much work is saved.

The processing of cache misses is complicated by the instability of wireless network connectivity. The transmission of requested data can suffer from long delays on weakly-connected links. A choice must be made as to whether the server should automatically transfer a full-sized file or let the user make the decision. Based on the importance of requested data, a user may be willing to wait for a long time. However, if each time a server has to consult the user, the server and the client will have to interact frequently. Therefore, in Coda the knowledge of user patience is required. Although the formula presented in Coda system research is extended in this thesis based on available mobility information, current modelling of user patience does not yet have a sound solution [MES95].

In summary, the issues of mobile transaction processing are attacked from different dimensions such as the mobile unit's role, replica and concurrency control, cache management, and resource optimisation. Fundamentally, efforts are made to balance three major aspects, i.e. network connectivity, resource availability, and data consistency. Nevertheless, the research in this area largely focuses attention on data management itself and does not exploit the benefits of mobility information. In contrast, the work in this thesis takes a broader viewpoint and attempts to incorporate mobility information into the design of applications.

Chapter 6

CONCLUSIONS

This chapter highlights the major contributions achieved by this work and describes the future directions for this research.

In this work two types of problems have been examined: *mobile environment modelling* and *mobile transaction processing.* The research in this thesis represents a substantial effort, though the result is still primary.

In order to better support mobile computing, the ability to capture and transmit mobile information should be taken into account as part of the design objectives in the next generation of networking and operating systems. The concept and representation of mobility developed in this thesis offers the groundwork leading to a theoretical foundation.

Meanwhile, although the effect of mobility of computers on transaction processing has been recognised, the work presented in this thesis has made a first attempt to apply mobility information to this domain. By viewing a mobile host and a fixed host as two relatively independent computing stations, the approach proposed in this thesis can support autonomous operations with better system performance.

6.1 Major Contributions

The major contributions of this work are as follows:

1. *Classification of Environment Factors:*

Although environment factors are important in developing adaptive applications, no systematic investigation has been reported. In this work, some environment factors are identified and a primary classification is presented. Based on the classification, the concept of mobility feature is defined.

2. *Mobility Information Measurement and Transformation:*

   The utilisation of mobility information relies on the quantisation of mobility features. In order to support decision-making in a mobile environment, an integrated mobility representing model, called unit mobility and unit relative mobility, is proposed. It provides a general way for efficient transmission, representation and transformation of mobility information.

3. *A General-purpose Adaptive Transaction Scheduling Approach:*

   Although the issues of transaction management in mobile computing have been recognised, the knowledge of mobility has not been explicitly applied to transaction processing in the data management area. To support autonomous processing with the existence of disconnection, a scheme characterised by a centralised transaction schedule and two-phases transaction commit is proposed. It presumes that mobility information with respect to disconnection ratio and bandwidth is available.

4. *Quantified Site/Policy Selection Criterion:*

   The kernel of the adaptive transaction scheduling approach is a decision-making criterion that allows the model to select the transaction processing policies/sites dynamically. This quantified criterion is not only assessing the performance of transaction processing in terms of communication cost, but also measuring the tolerance degree of a mobile user by utilising the concept of the patience time introduced in [MES95]. The measurement method of patience time is generalised in this work by introducing extra mobility factors.

5. *Simulation Model and Evaluation:*

The performance of proposed transaction scheduling approach is evaluated by the means of simulation. A queue model is designed and implemented in discrete-event simulation language (SIMSCRIPT). The comparisons between fixed policies and the proposed adaptive one are performed. The results of simulation have shown that the adaptive approach in general produces better performance, and confirms that mobility information is useful in designing better algorithms in a mobile distributed environment.

6.2 Future Directions

Since research and application in mobile computing are still in the early stages of development, many questions remain unanswered. Some directions that can be further explored include the following:

1. *Enhance the simulation model to reflect more realistic computing models.* In the current implementation of the simulation model, the effect of deadlock and cascade abort is not modelled. Moreover, the model does not fully reveal the impact of a client cache on data transmission, which is instead reflected with a random variable. Although reasonable data are used in the simulation experiments, they are not from practical applications. Therefore, it is desirable to collect more practical data to evaluate and improve the model.

2. *Study more efficient schemes of cache management.* As mentioned in the description of the ATS approach, how to estimate the read/write time bound for cached data is an open question. A decision making criterion should be developed in order to meet varying data consistency requirements. As a conjecture: mobility information may be useful to determine such a time bound.

3. *Explore the utilisation of mobility information in other application domains.* In this research, transaction processing is selected to be a particular domain for designing adaptive applications. Nevertheless, the utilisation of mobility information can be wider than this. Multimedia applications, for instance, in a

mobile environment can be another good candidate to benefit from mobility information.

4. *Investigate the methods of mobility data acquisition.* This research assumes that, for a given mobile factor, its data can be quantified and physically obtained in some way. For example, it is assumed that current bandwidth of network can be obtained by monitoring the communication port. However, it is unknown whether this is a feasible way for engineering implementation. To use mobility information in an actual system, this assumption must be validated. Therefore, more work should be done in this direction, probably joined with the researchers in engineering areas.

Bibliography

[ABB+96] A. Alwan, R. Bagrodia, N. BamBos, M. Gerla, L. Kleinrock, J. Short, and J. Villasenor. Adaptive mobile multimedia networks. *IEEE Personal Communications*, pages 34–51, April 1996.

[Ach95] A. Acharya. Structuring distributed algorithms and services for networks with mobile hosts. Ph.D. dissertation, the State University of New Jersey, New Brunswick, New Jersey, 1995.

[AIB94] A. Acharya, T. Imielinkski, and B.R. Bradrinath. Dataman project: Towards a mosaic-like location-dependant information service for mobile clients. Technical Report TR-93-320, Rutgers University, 1994.

[AK93] R. Alonso and H.F. Korth. Database system issues in nomadic computing. *ACM SIGMOD Record*, vol. 22:388–392, May 1993.

[Chr93] P.K. Chrysanthis. Transaction processing in mobile computing environment. In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 77–83, October 1993.

[CL91] M.J. Carey and M. Livny. Conflict detection trade-offs for replicated data. *ACM Transactions on Database Systems*, vol. 16:703–746, December 1991.

[DBCF94] N. Davies, G.S. Blair, K. Cheverst, and A. Friday. Supporting adaptive services in a heterogeneous mobile environment. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications* [1], December 1994.

[Duc92] D. Duchamp. Issues in wireless mobile computing. In *Proceedings of the 3rd Workshop on Workstation Operating Systems*, pages 2–10, Key Biscayne, FL, April 1992.

[EJB95] A.K. Elmargarmid, J. Jing, and O. Bukhres. An efficient and reliable reservation algorithm for mobile transactions. Technical Report CSDTR-95-018, Computer Science Department, Purdue University, 1995.

[EJF94] A. Elmagarmid, J. Jing, and T. Furukawa. Wireless client/server computing for personal information services and applications. In *Proceedings*

of Workshop on Mobile Computing Systems and Applications, December 1994.

[GA93] S. Ganguly and R. Alonso. Query optimisation for energy efficiency in mobile environments. Technical Report MITL-TR-48-93, Matasushia Information Technology Laboratory, 1993.

[IB92] T. Imielinski and B.R. Badrinath. Mobile wireless computing solutions and challenges in data management. Technical Report DCS-TR-296/WINLAB-TR-49, Computer Science Department, Rugers University, 1992.

[IV94] T. Imielinski and S. Viswanathan. Adaptive wireless information systems. In *Proceedings of SIGDBS Conference* [2], Tokyo, Japan, October 1994.

[JBE95] J. Jing, O. Bukhres, and A. Elmagarmid. Distributed lock management for mobile transactions. In *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, pages 118–25, Vancouver, BC, Canada, May 1995.

[JKT88] B.C. Jenq, W.H. Kohler, and D. Towsley. A queuing network model for a distributed database tested system. *IEEE Transactions on Software Engineering*, vol. 14(7):908–921, July 1988.

[JM96] D.B. Johnson and D.A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Communications*, vol. 3(1):34–42, February 1996.

[Kat94] R.H. Katz. Adaptation and mobility in wireless information systems. *IEEE Personal Communications*, vol. 1(1):6–17, 1994.

[KMV87] P.J. Kiviat, H. Markowitz, and R. Villanueva. *SIMSCRIPT II.5 programming language*. Los Angeles: C.A.C.I., 1987.

[KTW97a] J. Klingemann, T. Tesch, and J. Wäsch. Cooperative data management and its application to mobile computing. *Special Issue on CoopIS-97 of the International Journal of Cooperative Information Systems*, pages 34–51, April 1997.

[KTW97b] J. Klingemann, T. Tesch, and J. Wäsch. Enabling cooperatation among disconnected mobile users. In *Proceedings of the 2nd IFCIS International Conference on Cooperative Information Systems*, Kiawah Island, South Carolina, USA, June 1997.

[Lin94] Y. B. Lin. Determining the user locations for personal communications services networks. *IEEE Transaction on Vehicular Technology*, vol. 43(3):466–473, 1994.

[LM95] G. Y. Liu and G. Q. Maguire. A predictive mobility management scheme for supporting wireless mobile computing. Technical Report ISRN 1103-534X, Telecommunication Systems Laboratory, February 1995.

[MDC93] B. March, D. Douglis, and R. Caceres. System issues in mobile computing. Technical Report MITL-TR-50-93, Matasushia Information Technology Laboratory, February 1993.

[MES95] L.B. Mummert, M.R. Ebling, and M. Satyanarayanan. Exploiting weak connectivity for mobile file access. In *Proceedings of Symposium on Operating System Principles*, volume vol. 29, pages 143–155, Copper Mountain Resort, Colorado, December 1995.

[MHB97] M.H.Dunham, A. Helal, and S Balakrishnan. A mobile transaction model that captures both the data and movement behaviour. In *ACM special Issues on mobility of syttems, users, data and computing*, volume vol. 2, pages 149–162, 1997.

[Nar94] V.R. Narasayya. Distributed transactions in a mobile computing system. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications* [1], Santa Cruz, CA, USA, December 1994.

[Nie95] J. D. Nielsen. Transactions in mobile computing. Master thesis, Universidade do Minho, Portugal, 1995.

[NPS95] B.D. Noble, M. Price, and M. Satyanarayanan. A programming interface for application-aware adaptation in mobile computing. Technical Report CMU-CS-95-119, Computer Science Department, Carnegie Mellon University, February 1995.

[OV91] M.T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Englewood Cliffs, N.J.: Prentice Hall, 1991.

[PB93] E. Pitoura and B. Bhargava. Dealing with mobility: Issues and research challenger. Technical Report CSD-TR-93-070, Purdue University, West Lafayette, IN, November 1993.

[PB94] E. Pitoura and B. Bhargava. Building information systems for mobile environments. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pages 371–378, November 1994.

[PB95] 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, May 1995.

[PO95] R. J. Peters and M. T. Özsu. Relative mobility: Capturing dynamics in mobile computing. In *unpublished manuscript*, 1995.

[RB95] S. Rajagopalan and B.R. Badrinath. An adaptive location management strategy for mobile ip. In *Proceedings of the 1st International Conference on Mobile Computing Networking*, November 1995.

[SAW94] B. Schilit, N. I. Adams, and R. Want. Context-aware computing applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications* [1], pages 85–90, Santa Cruz, CA, USA, December 1994.

[Sto94] M. Stonebraker. *Readings in Database Systems*, chapter 7, pages 548–566. Morgan Kaufmann Publishers, Inc., 2nd edition, 1994.

[Tho96] A. Thomasian. *Database Concurrency Control: Methods, Performance, and Analysis*. Kluwer Academic Publishers, 1996.

[VB94] G.M. Voelker and B.N. Bershad. Mobisaic: An information system for a mobile wireless computing environment. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications* [1], Santa Cruz, CA, USA, December 1994.

[WC95] G.D. Walborn and P.K. Chrysanthis. Supporting semantics-based transaction processing in mobile database applications. In *Proceedings of the 14th IEEE Symposium on Reliable Distributed Systems*, pages 31–40, September 1995.

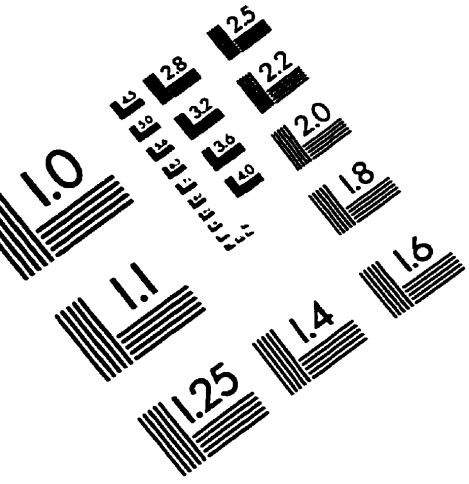
[Yua93] R. Yuan. An adaptive routing scheme for wireless mobile computing. In *Proceedings of the 4th WINLAB Workshop on 3rd Generation Wireless Information Networks*, pages 55–62, October 1993.

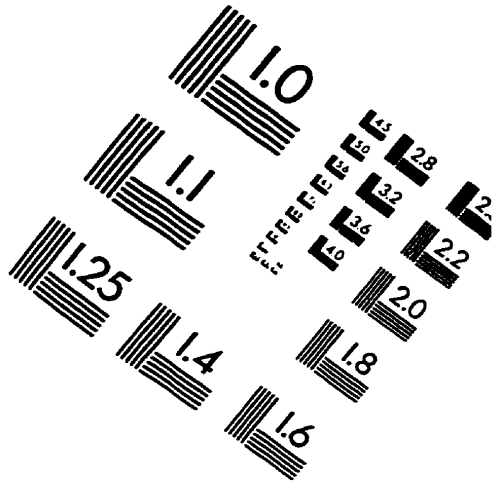
[YZ94a] L.H. Yeo and A. Zaslavvsky. Layered approach to transaction management in multidatabase systems. In *Proceedings of the 5th International HongKong Computer Society Database Workshop: Next Generation Database Systems*, pages 179–189, 1994.

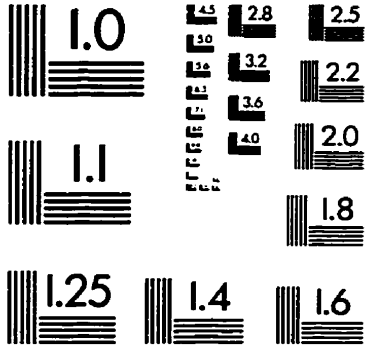
[YZ94b] L.H. Yeo and A. Zaslavvsky. Submission of transactions from mobile workstations in a co-operative multidatabase processing environment. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 372–379, Poznan, Poland, June 1994.

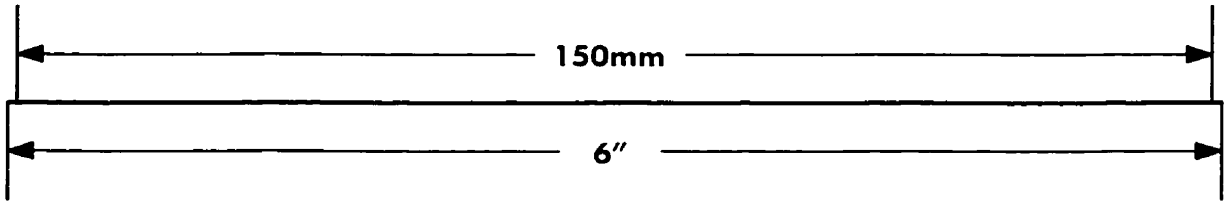
[1] available at http://snapple.cs.washington.edu/mobile/mcsa94
[2] available at ftp://paul.rutgers.edu/pub/badri/awis.ps.Z

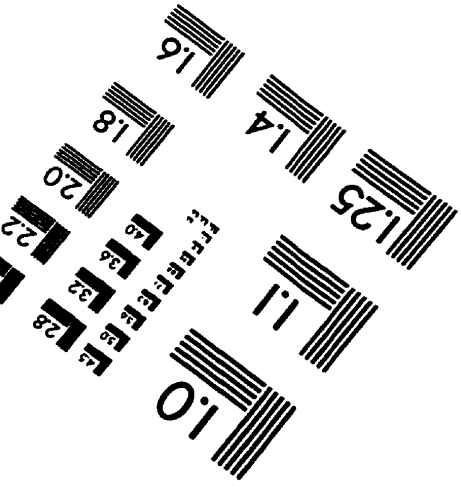
IMAGE EVALUATION
TEST TARGET (QA-3)

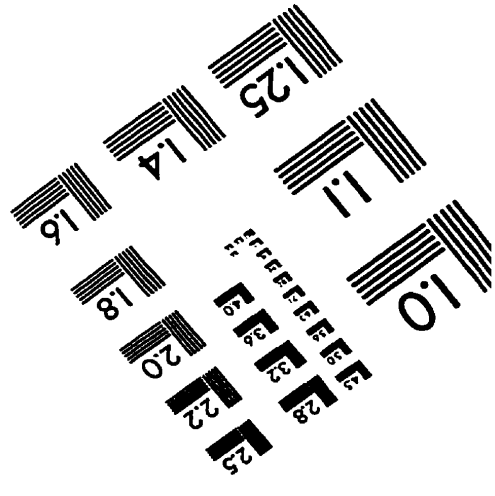












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989