NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

MOBILE COMPUTING MODEL

Graduation Project

COM-400

Student: Faisal Rehman Mir(20002126)

Supervisor: Dr.Jamal Fathi

NICOSIA-2006

ACKNOWLEDGEMENTS

First of all i would like to thank God,enabled me to complete my project and also my bsc degree.i am also grateful to my parents especially,my dear father Aman Ullah Mir who has been hard working abroad for me to fulfill his desire of my computer engineering degree.they have supported me financially at anywhere in the world and also thier pray so,i get to success in the matter of education.

I would also express my thanks to my dear aunt's Mrs.Shamshad Aqeel who supported me and also boosted up my studies.

I want to say my thanks to my dear brother Mr.Mohammad Aman Mir who encouraged and assisted me in every phase of my life however,i would like to say my thanks to thier friends,Mr. Mohammad Jawad,Mohammad Arshad Khan and Abdul Razzaq Mr.Nadeem Butt Mr.Mohsan Khant.They encouraged me alot to complete my project.i would to thank them as they contributed their time and provided me a helpful suggestions about my graduation project.

Finally,i would like to extend my warm welcome and thanks to my dear supervisor Dr.Jamal Fathi,without whom this project would have not been possible.whose words of encouragement keep my work more effected.his faith and integrety of his invaluable knowledge to keen me intersted in my this project.he is an excellent and elite advisor and also a best teacher.

CONTENTS

1. MOBILE COMPUTING

1.1 Overview

In this chapter we are going to discuss mobile computing as the introduction of mobile computing data which is included input section and execution section and also text entry

For explain who many ways we can enter the data into the mobile by using key pad and touching pad.

1.2 Introduction

Among the earliest of handheld devices was the HP95LX, which was released in 1991 by *Hewlett Packard. The technological equivalent of an IBM XT shrunk into a clam-shell* format; the HP95LX was small enough to fit in the palm of one's hand. Although the term Personal Digital Assistant (PDA) had not yet entered the vernacular to describe a handheld computer, this was the first PDA. The HP95LX provided a small QWERTY keyboard for text entry, although touch-typing was impossible due to its size. Later devices followed. These devices demonstrated that the QWERTY keyboard could be adapted to mobile computing devices.

The early 1990s was an exciting time for mobile computing, due to the arrival of pen computing. The ideas touted much earlier by Kay and Goldberg in their Dynabook project finally surfaced in commercial products. However, the initial devices were bulky, expensive, and power hungry, and they could not deliver in the one area that garnered the most attention – handwriting recognition. Without a keyboard, the pen was the primary input device. If only "selecting" and "annotating" were required, then the success of pen entry seemed assured. However, some applications demanded entry of text as machine-readable characters, and the handwriting recognition technology of the time was not up to the challenge. Products from this era, such as GridPad, Momenta, Poquet, and PenPad, did not sustain the volume of sales necessary for commercial viability. Most endured only a year or two.

announcement that it would enter the pen computing market. Thus emerged the Apple MessagePad . Apple was a major player in desktop computing at the time, and its commitment to pen computing was taken seriously.

To a certain extent, Apple added legitimacy to this entire segment of the computing market. However, the Newton was expensive and rather specialized. It was embraced by many technophiles but it did not significantly penetrate the larger desktop or consumer market. The Newton's handwriting recognition, particularly on early models, was so poor that is was ridiculed in the media, for example by Garry Trudeau in his celebrated Doonesbury cartoons. Nevertheless the Newton received considerable attention and it ultimately set the stage for future mobile devices. The next significant event in mobile or pen computing was the release the Palm Pilot by Palm Inc. The Palm was an instant hit. Five years hence, it is the technology of choice for millions of users of mobile devices. There is much speculation on why the Palm was so successful. Some factors seem relevant: a few hundred less than a Newton. The Palm supported HotSync as a standard feature. The Palm was smaller and lighter than the Newton, and could fit in one's pocket. Because of lower power consumption, the batteries lasted for weeks instead of hours. Finally, and perhaps most importantly, the Palm avoided the thorny issue of cursive or block-letter handwriting recognition by introducing a greatly simplified handwriting technique known as Graffiti. By simplifying recognition, Graffiti required less CPU power and memory, achieved better character recognition, and ultimately enjoyed widespread acceptance among users. The year 1996 also saw the release of the Windows CE operating system by Microsoft .Devices such a Casio's Cassiopeia or Philips' Velo, which used Windows CE, were more powerful than previous mobile computing devices, but were also larger. The first version of Windows CE only supported a soft keyboard for text entry, but later versions included the Jot handwriting recogniser, by Communications Intelligence Corp. A recent entry in the pen computing market is the Crosspad by A. J. Cross Company.

The Crosspad avoids handwriting recognition by recording the user's writing as ink trails. The user's notes are downloaded to a desktop computer for storage and subsequent recognition on the desktop computer. Software accompanying the Crosspad supports handwriting recognition of keywords, and indexing and retrieval by keyword. All of the devices mentioned above are handheld computers that support text entry. Another, quite different, group of devices that support text entry are messaging devices such as mobile phones and pagers. In Europe, where text messaging has been available more text messages are transmitted daily than voice messages. In North America most mobile phones and pagers do not yet support text messaging but this is changing. The latest generation of two-way pagers such as the Blackberry by Research In Motion and PageWriter by Motorola support text entry via a miniature Qwerty keyboard. We conclude this perspective with a hint at what the future might hold. At the top of our list is a device combining the programmability of the PDA, wireless telephony, text messaging, and unfettered internet and email access. Pieces of this scenario already exist, but implementations require a specialized configuration, optional components, or support only a subset of standard features. We view these as transitional technologies. Devices that do not quite make the grade, in our view, are those that require an add-on RF transceiver, or provide internet access only to sites supporting a specialized protocol (e.g., wireless access protocol, or, WAP). For text input, the pen-based paradigm has dominated the PDA market, but there is a parallel trend toward text messaging in mobile phones and pagers using keyboard-based technology. If these technologies converge, then which text input technology will prevail?

1.3 Text Entry

There are two competing paradigms for mobile text input: pen-based input and keyboard-based input. Both emerged from ancient technologies ("ancient" in that they predate computers): typing and handwriting. User experience with typing and handwriting greatly influences expectations for text entry in mobile computing; however the two tasks are fundamentally different. A key feature of keyboard-based text entry is that it directly produces machine-readable text (i.e., ASCII characters), a necessary feature for indexing,

searching, and handling by contemporary character-based technology. Handwriting without character recognition produces "digital ink". This is fine for some applications such as annotation, visual art, and graphic design. However, digital ink requires more memory and in general it is not well managed by computing technology. Specifically, digital ink is difficult to index and search report some success with a graphical search mechanism for digital ink that is not based on recognition). For handwritten text entry to achieve wide appeal, it must be coupled with recognition technology.

An important consideration implicit in the discussion of text input technology is user satisfaction. The point was made earlier that the Palm succeeded where the Newton failed, in part because of users' acceptance of Graffiti as a text input technology. Users' expectations for text entry are set by current practice. Touch typing speeds in the range of 20 - 40 words per minute (wpm) are modest and achievable for hunt-and-peck typists. Rates in the 40-60 wpm range are achievable for touch typists, and with practice, skilled touch typists can achieve rates greater than 60 wpm. Handwriting speeds are commonly in the 15-25 wpm range. These statistics are confirmed by several sources. Users, perhaps unrealistically, expect to achieve text input rates within these ranges on mobile devices. Furthermore, they expect these rates immediately, or within a short time of using a new input technology.

1.4 Summary

In this chapter we fond and we discussed form the binging who may ways we can entered the data into the mobile system. We mentioned some methods. And also we did consideration to implicit in the discussion of text input technology for use satisfaction. And about input entry we mentioned two methods and in these two methods we have some branches who obey the rule of these methods.

2. EVALUATION

2.1 Overview

In this chapter we are going to discuss about evaluation to keep information of text entery and repersent some technologies which are refined with input entry.and also discuss about methodogy to generalizable for the results. Text Creation versus Text Copy for distinction in text entry evaluations is between text creation and text copy. Novice versus Expert Performance for the the design of text input methods focuses on the potential, or expert, text entry rate of a particular design.

2.2 Introduction

Research in mobile text entry is flourishing in part because user needs are not currently met. Typically, traditional text inputs technologies are refined or new input technologies are invented. Either way, evaluation is a critical and demanding part of the research program. The questions researchers pose are ambitious: "Can entry rates be improved if we arrange the buttons on a keyboard in a certain way?" or "What is the effect if we use context to guess the next letter or word?" or "Can we apply an altogether different technology, like pie menus, touch pads, or pattern recognition, to the problem of text input?" In this section,

2.3 Methodology

An evaluation is valuable and useful if the methodology is reproducible and results are generalizable. "Reproducible" implies that other researchers can duplicate the method to confirm or refute results. This is achieved for the most part simply by following an appropriate reporting style "Generalizable" implies that results have implications beyond the narrow context of the controlled experiment. This is achieved through a well-designed experiment that gathers measures that are accurate and relevant, in tasks that are representative of real-life behavior. There is, unfortunately, a trade off here. In real-life,

people rarely focus solely on a single task. Methodologies so designed, therefore, may find that the measurements include behaviors not specifically required of the interaction technique. The trade off, therefore, is between the accuracy of our answers and the importance and relevance of the questions they seek to address. That is, we can choose between providing accurate answers to narrow questions, or providing vague answers to broad questions.

The reader is implored not to interpret this too strictly, but, hopefully, the point is made. In designing an experiment, we strive for the best of both worlds; answering interesting or broad questions (viz. using real-life tasks) and doing so accurately (viz. accurately measuring the behavior of interest, such as entry speed or accuracy). In the following sections, we identify some factors relevant to methodologies for evaluating text entry on mobile systems.

2.4 Text Creation versus Text Copy

An important distinction in text entry evaluations is between text creation and text copy. In a text copy task, the subject is given text to enter using the input technique under investigation. As a backdrop for discussing these two types of tasks, we introduce the term focus of attention (FOA). FOA speaks to the attention demands of the task. Consider the case of an expert touch-typist using a Qwerty keyboard to copy text from a nearby sheet of paper. This is a text copy task. Because the typist is an expert, she does look at the keyboard or display she attends only to the source text. This is a single FOA task. However, if input is via a stylus and soft keyboard, the typist must also attend to the keyboard. (A soft keyboard cannot be operated "eyes free".) Stylus typing, therefore, is a two FOA task. If the typist is less-than-expert and corrects errors, she must look at the display to monitor results. This increases touch typing to a two FOA task, and stylus typing to a three FOA task. Clearly, the feedback channel is overburdened in a three FOA task. Despite the additional FOA, text copy tasks are generally preferred to text creation tasks for empirical evaluations. There are several reasons. One is the possible presence of behaviours not required of the interaction technique. Examples include pondering ("What

should I enter next?") or secondary tasks (e.g., fiddling with system features). Clearly measurement of text entry speed are compromised if such behaviours are present.

A second difficulty with text creation tasks is identifying errors it is difficult to know exactly what a subject intended to enter if the subject is generating the text. Even if the message content is known a priori, errors in spelling or memory recall may occur, and these meta-level mistakes are often indistinguishable from errors due to the interface itself. A third difficulty is the loss of control over the distribution of letters and words entered. The task should require the subject to enter a representative number of occurrences of characters or words in the language (i.e., results are generalizable). However, it is not possible to control for this if the subject is generating the text. The main advantage of a text creation task is that it mimics typical usage. The disadvantages just cited, however, are significant and drive most researchers to use text copy tasks despite the increased FOA. One way to mitigate the effects of increased FOA is to dictate the source text through the audio channel. Ward and colleagues used this technique, however they noted that subjects found the approach stressful and hard to follow. A carefully designed experiment may capture the strengths of both a text creation task and a text copy task. One technique is to present subjects with short, easy-to-memorize phrases of text. Subjects are directed to read and memorize each phrase before entering it. Entry proceeding thus, benefits from the desirable property of a text creation task, namely, reduced FOA. As well, the desirable properties of a text copy task are captured, that is, control over letter and word frequencies, and performance measurements that exclude thinking about what to write. There are numerous examples of this approach in the literature.A similar technique is to present text in large a block (e.g., a complete paragraph) but to interleave each line of the presented text (input) with each line of generated text (output). As input proceeds, each character entered appears directly below the intended character. This is a text copy task, however FOA is reduced to that of a text creation task, because subjects attend only to one location for both the source text and the results of entry.

2.5 Novice versus Expert Performance

Most work on the design of text input methods focuses on the potential, or expert, text entry rate of a particular design. However, the novice experience is paramount for the success of new text input methods. This is at least partially due to the target market. Mobile devices, such as mobile phones and Pads, once specialized tools for professionals, are increasingly targeted for the consumer market. It follows that "immediate usability" is important. In other words, it may be a moot point to establish the expert, or "potential" text entry rate for an input technique if prolonged practice is required to achieve it. Consumers, discouraged by their initial experience and frustration, may never invest the required effort to become experts.

However, measuring immediate usability is easier said than done. In typical studies of new interaction techniques, participants are given a demonstration of the technique followed by a brief practice session. Then, data collection proceeds over several blocks of trials. However, the measurements are a poor indicator of novice behaviour, at least in the sense of immediate, or walk-up, usability. Within a few minutes, participants' knowledge of the interaction technique develops and the novice status fades. Measuring expert performance is also not easy, since acquisition of expertise requires many blocks of trials administered over many days, or more. Some longitudinal text entry studies evaluations are hereby cited.An example of results from a typical longitudinal study is given in Figure 2.1. Users' improvement in entry speed is shown over 20 sessions of input for two types of soft keyboards. The data were fitted to the standard power law of learning. Prediction equations and squared correlations are shown, as are extrapolations of the predictions to 60 sessions. Prediction equations and squared correlations are shown, as are extrapolations of the predictions to 60 sessions.

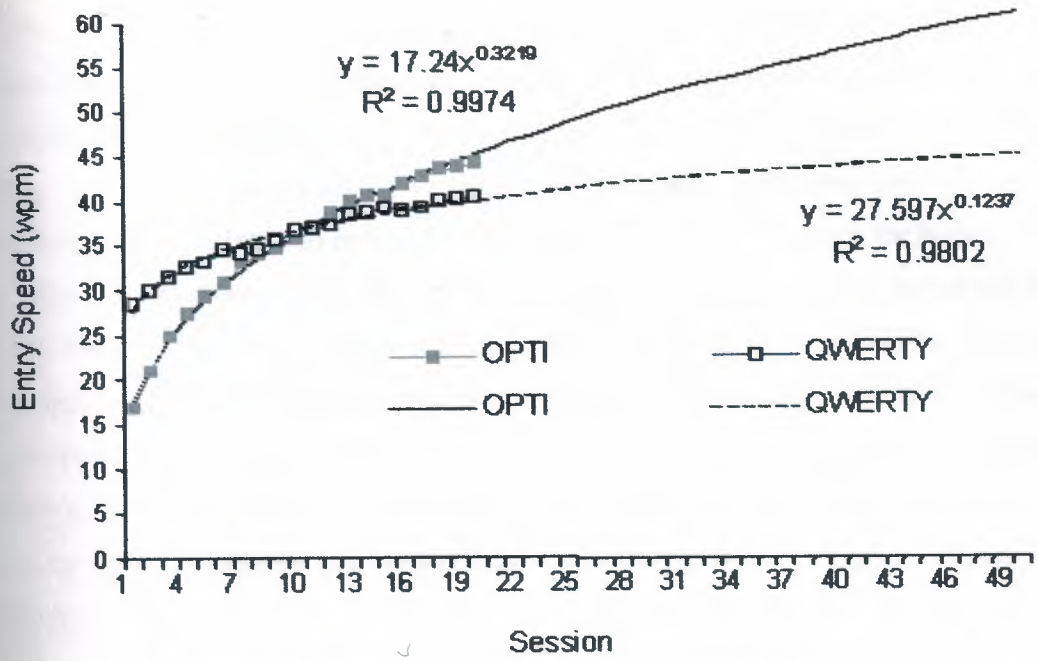

Figure 2.1. Reporting example for a longitudinal study

2.6 Quantitative Versus Qualitative Analyses

We noted earlier a trade off between the accuracy of answers and the relevance of the questions they seek to address. Quantitative evaluations tend to provide accurate answers to narrow questions, whereas qualitative evaluations tend to provide rather loose answers to broad but very important issues. Of course, researchers strive for the best of both worlds. In quantitative evaluations, "representative tasks" and "relevant measures" are used to ensure interesting or relevant questions are answered. In qualitative evaluations, robust test instruments are developed to ensure the answers are accurate, relevant, reproducible, and generalizable.When reporting quantitative results, there are many common pit-falls to avoid such as inaccuracy in measurements, lack of control or baseline conditions, inferring too much from data, using too small a sample size, collecting insufficient data, artificially biasing data by aggregation, non-random presentation of conditions, or inappropriate treatment of outliers. The reader is directed to textbooks in experimental psychology for further discussions. For a discussion on aggregation bias, see Walker. Researchers may be excused for slightly bending the rules,

perhaps, but all too common are published reports stating only qualitative results steeped in anecdote, or, worse yet, testimonials unsupported by empirical data. An excerpt from *one such publication illustrates our point: While we have yet not done systematic user* testing, anecdotal experience to date is consistent: Users well practiced in both and consistently find the latter to be about three times faster, with accuracy for both systems very high. Testimonials such as this are of questionable merit; they surely do not meet the criteria for good research that results are generalizable and reproducible. Unless a controlled experiment is performed using quantitative metrics or established qualitative test instruments, there is no way to gauge the performance of a new text input technique. Conjuring up a new input technique is fine, but research demands more. It demands that new ideas are implemented and evaluated in conformance with the rigors of an empirical evaluation.

Although quantitative tests form the backbone of any scientific study, qualitative aspects of the investigation are also important. In human-computer interfaces, users must feel comfortable with the interaction technique and must feel their efforts have a reasonable payoff in their ability to accomplish tasks. Participants will develop impressions of each device or condition tested, and these should be solicited and accounted for in the final analysis. Typically, these opinions are sought via questionnaire, administered at the end of a condition or experiment. The reader is referred to textbooks in human-computer interaction for direction in questionnaire design.

2.7 Speed

For text input there are two primary evaluation metrics: speed and accuracy. The simplest way to measure and report speed is to measure the number of characters entered per second during a trial, perhaps averaged over blocks of trials. This gives a measure in characters per second. To convert this to words per minute the standard typists' definition of a word as five characters is employed. Therefore, words per minute are obtained by multiplying characters per second by 60 and dividing by 5.

2.8 Accuracy

Accuracy is more problematic. For a simple treatment of accuracy, we obtain a metric that captures the number of characters in error during a trial, and report these as a percentage of all characters in the presented text. A more complete analysis involves determining what kind of errors occurred, and why. The difficulty arises from the compounding nature of mistakes and the desire to automate as much of the data measurement and analysis as possible. Four basic types of errors include entering an incorrect character omitting a character adding an extra character or, swapping neighboring characters. While it is straightforward for a human to compare the intended text with the generated text and tabulate the errors, in practice the amount of analysis is simply too much, given a reasonable number of subjects, conditions, and trials. Additionally, tabulation errors may be introduced if performed manually. However, automating error tabulation is not trivial. Consider an experiment where the subject is required to enter the 19character phrase: "the quick brown fox". If the subject enters "the quxxi brown fox", the incorrect word contains either three substitution errors, or two insertion and two omission errors. The explanation with the fewest number of errors is preferred, and, in this simple example,Algorithms for "string distance" calculations, such as the Levenshtein string distance statistic, might assist in automating analyses such as these, as demonstrated by Soukoreff and MacKenzie.

A useful tool for designers is the confusion matrix, graphically depicting the frequency of character-level transcription errors. Figure 2.2 is a confusion matrix taken from Chang and MacKenzie's comparative study of two handwriting recognisers. The confusion matrix displays intended characters versus recognized characters illustrating how often an intended character was misrecognised and interpreted as another character. Each dot represents three occurrences.

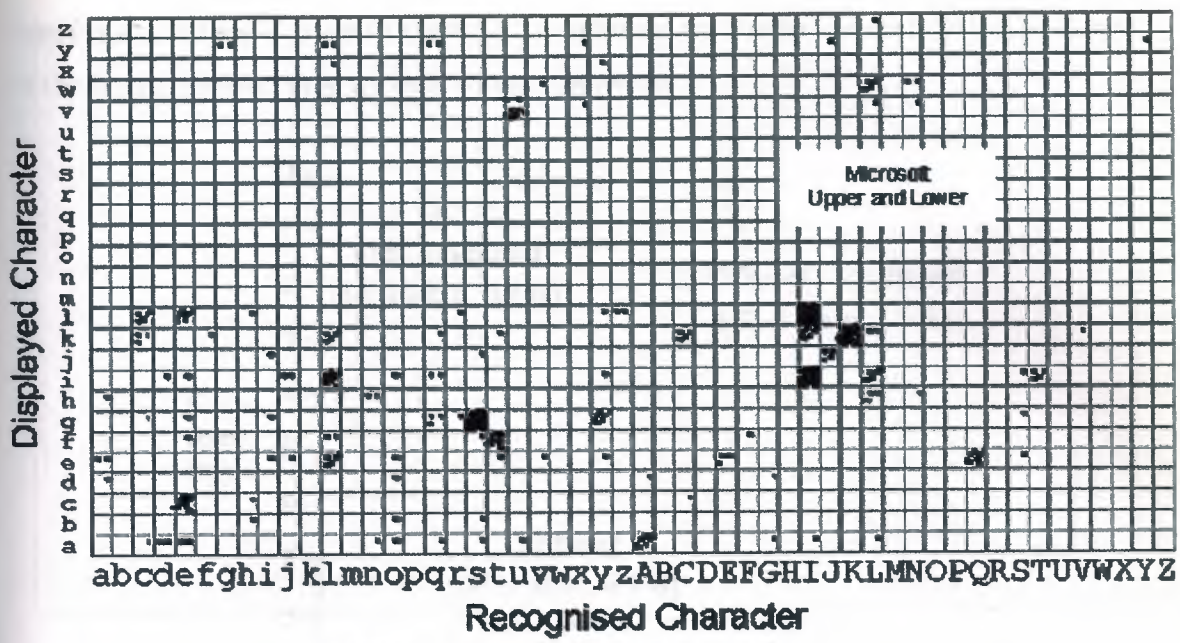

Figure 2.2 Sample Confusion Matrixes

Difficulties in error tabulation have pushed some researchers to ignore errors altogether, or to force the subject to enter correct text only. Directing subjects to "correct as you go" is another possible approach. Assuming subjects adhere to instructions, the resulting text is error free; thus, the error rate is 0%. In general though, subjects will leave errors in the generated text, even if requested not to. The result is two levels of errors: those that were corrected, and those that were not. For the corrected errors, overhead is incurred in making the corrections. A reasonable measure of accuracy in this case is keystrokes per *character*. For a Qwerty keyboard, the ideal is KSPC = 1.0, but, in practice, KSPC > 1 if subjects correct as they go. If, for example, a 25-character phrase was entered and two substitution errors occurred, each corrected by pressing BACKSPACE followed by the correct character,

Clearly, both speed and accuracy must be measured and analysed. Speed and accuracy are well known to exist in a continuum, wherein speed is traded for accuracy and vice versa. Subjects can enter text more quickly if they are willing to sacrifice accuracy. For subjects to perform with high accuracy, they must slow down. The trade-off suggests that measuring only speed or only accuracy will skew the results so as to make the text input method appear better than it really is. An example of a reporting technique that combines

speed and accuracy is given in Figure 2.3 Conditions are "better" toward the top and right of the figure, because they are both fast and accurate

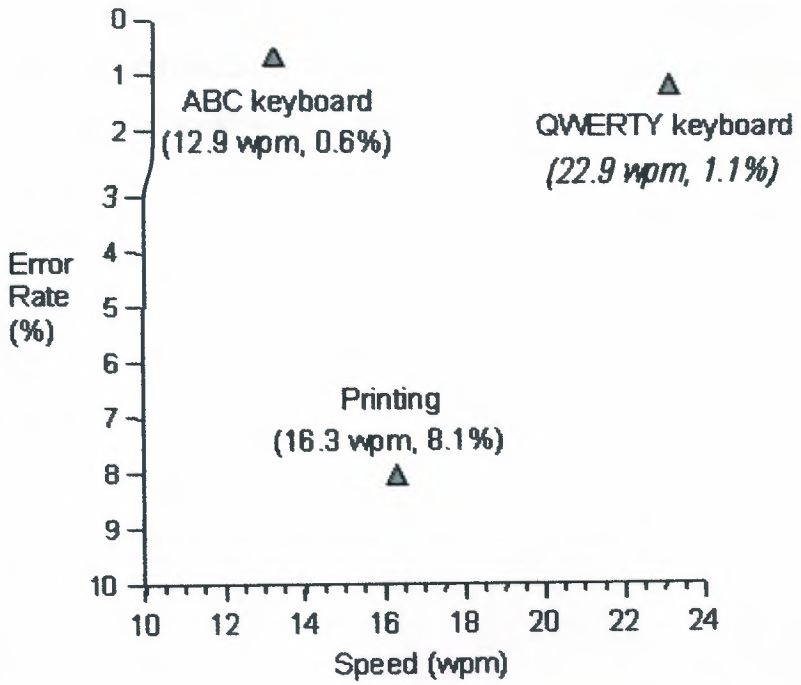

Figure 2.3. Simultaneous presentation of results for speed and accuracy

2.9 Other Factors

The text input process can be significantly impacted by factors bearing little on the input device, such as whether the device is operated standing, sitting, or walking, or whether it is operated with one or two hands. Designers of novel text input techniques must be aware that users want to operate mobile devices anytime, anywhere. Lack of a one-hand interaction method may impact the commercial success of a technology. Evaluations are often conducted to test a refinement to existing practice. Often the new technique is an improvement over the status quo. Some key initiatives in improving current practice through language and movement modelling are presented in the next section.

2.10 Summary

According to the calculation of input entry we did in this chapter some calculation by using several ways. We did some analysis to find out the result in this section characters are also included. By the calculation we did or we are care of the speed and the Accuracy. We did our calculation by graphically to get the results.

3. Locating Objects in Mobile Computing

3.1 Overview

Approaches to storing location information range between two extremes. At one extreme, up-to-date information of the exact location of all users is maintained at each and every network location. In this case, locating a user reduces to querying a local database. On the other hand, each time the location of a user changes, a large number of associated location databases must be updated. At the other extreme, no information is stored at any site of the network. In this case, to locate a mobile user a global search at all network sites must be initiated. However, when a user moves, there is no cost associated with updating location databases. Between these two extremes, various approaches that balance the cost of lookups against the cost of updates are plausible. These approaches compromise the availability, precision or currency of the location information stored for each user (Figure 3.1). In terms of availability, choices range between saving the location at all network sites to not storing the location at all. In between these two approaches, location information may be maintained selectively at septic network sites. There is a wide range of selection criteria for the sites at which to save location information for each user. For example, a choice may be to save the location of users at the sites of their frequent callers. Imprecision in location information takes many forms. For instance, instead of maintaining the exact location of the user, a wider region or a set of possible locations is maintained. Currency refers to when the stored location information is updated. For instance, for highly mobile users it may make sense to defer updating the stored information about their location every time the users move. When current and precise information about a user's location is not available locally, locating the user involves a combination of some search procedure and a number of queries posed to databases storing locations.

3.2 Location Management

In mobile computing, mobile objects, e.g., mobile software or users using wireless hardware, may relocate themselves from one network location to another. To enable the ancient tracking of mobile objects, information about their current location may be stored at septic network sites. In abstract terms, location management involves two basic operations, lookups and updates. A lookup or search is invoked each time there is a need to locate a mobile object, e.g., to contact a mobile user or invoke mobile software. Updates of the stored location of a mobile object are initiated when the object moves to a new network location. In the rest of this section, we rest provide an overview of the problem and then introduce network architectures that are commonly associated with mobile computing.

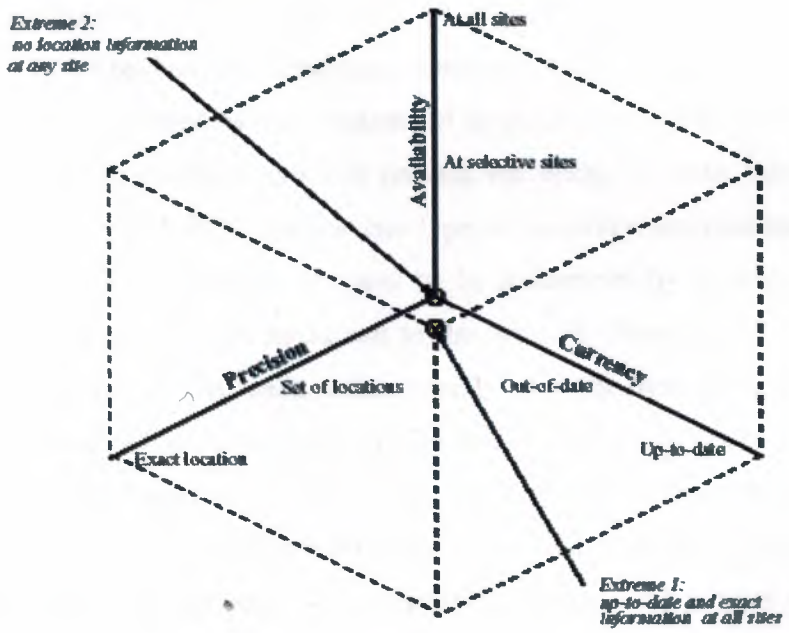

Figure 3.1 Approaches to Saving Location Information

3.2.1 Underlying Network Architecture

The networking infrastructure for providing ubiquitous wireless communication coverage is riper- scented by the personal communication system (PCS) also known by a number of deferent names such as personal communication network (PCN) and UMTS (universal

mobile communication system). While the architecture of the PCS has not evolved yet, it is expected that it will be partially based on the existing digital cellular architecture (see Figure3.2). This network conjuration consists of axed backbone networks extended with a number of mobile hosts (MHz) communicating directly with stationary transceivers called mobile support stations (MSS) or base stations. The area covered by an individual transceiver's signal is called a cell. The mobile host can communicate with other units, mobile or axed, only through the base station at the cell in which it resides. Thus to communicate with a mobile user, the base station of the cell in which it currently resides must be located. As a mobile host moves, it may cross the boundary of a cell, and enter an area covered by a deferent base station. This process is called hand and may involve updating any stored location information for the mobile host. It is speculated that ubiquitous communications will be provided by PCS in a hybrid fashion: heavily populated areas will be covered by cheap base stations of small radius (Pico cells); less populated areas will be covered by base stations of larger radius; and farm land, remote areas and highways with satellites that will provide the bridge between these deferent islands of population density. PCSs involve two types of mobility: terminal and personal mobility Terminal mobility allows a terminal to be indentured by a unique terminal denier independent of its point of attachment to the network. Personal mobility allows PCS users to make and receive calls independently of both their network point of attachment and a septic PCS terminal. Each mobile user explicitly registers itself to notify the system of its current location. The granularity of a registration area ranges from that of a single cell to a group of cells. Once the registration area is indentured, the user can be tracked inside this area using some form of paging. Paging is the process whereby to locate a mobile user, the system issues polling signals in a number of likely locations. By changing the size of a registration area, the edibility of any combination of registration and paging is attained If not explicitly stated otherwise, we use the term cell or zone as synonyms with registration area to indicate a uniquely deniable location where a mobile user can be found. In the cellular architecture, three levels are involved: the access, the axed, and the intelligent network the axed network is the wired backbone network. The access network is the interface between the mobile user and the axed network. The

intelligent network is the network connecting any location registers, i.e., registers used to store information about the location of mobile users.

This network is used to carry track related to tracking mobile users. The Signaling System and its signaling network is a good candidate to carry the signaling track in the intelligent network. Location management is handled at the data link or networking layer transparently from the layers above it each time a call is placed or a change in the network point of attachment occurs. Location management is an issue presents at all wireless networks (e.g., cellular, wireless LANs, and satellites). Although most solutions so far relate to cellular architectures at the data link Layer and to wireless LAN architectures at the networking layer, most are general enough to be applicable to deferent layers and architectures. In addition to handling mobility at lower layers, the need for information about the location of moving objects is encountered at the application level as well.

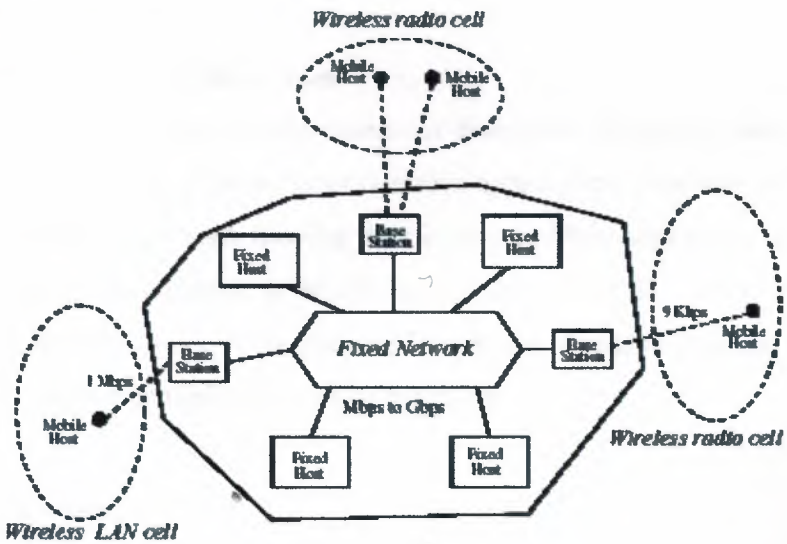

Figure 3.2 Wireless Computing Architecture

Applications may need information about the location of mobile users to answer a Variety of queries that involve location Other applications may involve updating environmental parameters and selecting locally available computing resources (e.g., nearest printer) There is no standard way for applications to acquire and use location

information. For example, applications may choose to maintain their own data structures for storing location information. The cellular architecture is not the sole infrastructure for wireless mobile computing. In its absence, various techniques may be employed to identify the current location of mobile users, for instance, users may be equipped with a Global Positioning System (GPS).GPS are space-based radio positioning systems that provide three-dimensional position, velocity and time information to suitably equipped users anywhere on or near the surface of the Earth. Common applications in this area include digital battle ends in the military context and transportation systems in the civilian industry. Finally, besides mobility tied to wireless hardware, the techniques presented in this paper are also applicable when the objective is to locate mobile code and data. Furthermore, similar techniques are also necessary when instead of location; the objective is to anciently retrieve other parole information related to mobile users. This information may include Quos related parameters or services.

3.3 Architectures of Location Databases

In this section, we describe basic architectures for distributed databases used for storing the location of moving users. The two most common approaches are a two-tier scheme in which the current location of each moving user is saved at two network locations and a tree-structured distributed database in which space is hierarchically decomposed in sub regions. We also describe a graph theoretic approach that employs regional directories. Finally, we refer brie to a centralized database approach.

3.3.1 Two-tier Schemes

In two-tier schemes, a home database, termed Home Location Register (HLR), is associated with each mobile user. The HLR is located at network location (zone) pre specie for each user. It maintains the current location of the user as part of the user's parole. The search and update Procedures are quite simple. To locate a user x, ax's HLR is indentured and queried. When a user x moves to a new zone, ax's HLR is contacted and updated to maintain the new location. As an enhancement to the above scheme, Visitor Location Registers (VLSI) is maintained at each zone. The VLR at a zone stores copies of paroles of users not at their home location and currently located inside that

zone. When a call is placed from zone I to user x, the VLR at zone I is queried rest and only if the user is not found there, is ax's HLR contacted. When a user x moves from zone I to j, in addition to updating ax's HLR, the entry for x is deleted from the VLR at zone I, and a new entry for x is added to the VLR at zone j. The two prevailing existing standards for cellular technologies, the Electronics Industry Association Telecommunications Industry Associations (EIA/TIA) Interim Standard commonly used in North America and the Global System for Mobile Communications (GSM) used in Europe, both support carrying out location strategies using Hers and VLSI at the Internet networking level, mobile IP is a medication to wire line IP that allows users to continue to receive messages independently of their point of attachment to the Internet. Mobile IP is designed within the IETF (Internet Engineering Task Force) and is outlined in a number of Requests for Comments (RFC).Wire line IP assumes that the network address of a node uniquely indentures the node's point of attachment to the Internet. Thus, a node must be located on the network indicated by its IP address to receive messages destined to it. To remedy this, in mobile IP, there are two IP addresses associated with each mobile node. One address, known as the home address of the node, is used to identify the node and is treated administratively just like a permanent IP address. When away from its home network, a care-of-address is associated with the mobile node and re acts the mobile node's current point of attachment. The care-of-address is either the address of a foreign agent which is a router on the visited network that provides services to the mobile node or a co-located address which is an address temporarily acquired by the mobile node. When a mobile node is away of its home, it registers its care-of-address with its home address. Then to deliver any messages, the home agent tunnels them to the care-of-address. One problem with the home location approach is that the assignment of the home register to a mobile object is permanent. Thus, long-lived objects cannot be appropriately handled, since their home location remains axed even when the objects permanently move to a deferent region. Another Drawback of the two-tier approach is that it does not scale well with highly distributed systems where sites are geographically widely dispersed. To contact an object, the possibly distant home location must be contacted rest. Similarly, even a move to a nearby location must be registered at a

potentially distant home location. Thus, locality of moves and calls is not taken advantage off.

3.3.2 Hierarchical Schemes

Hierarchical location schemes extend two-tier schemes by maintaining a hierarchy of location databases. In this hierarchy, a location database at a higher level contains location information for users located at levels below it. Usually, the hierarchy is tree-structured. In this case, the location database at a leaf serves a single zone (cell) and contains entries for all users registered in this zone.

A database at an internal node maintains information about users registered in the set of zones in its sub tree. For each mobile user, this information is either a pointer to an entry at a lower level database or the user's actual current location. The databases are usually interconnected by the links of the intelligent signaling network, e.g., a Common Channel Signaling (CCS) network. For instance, in telephony, the databases may be placed at the telephone switches. It is often the case that the only way that two zones can communicate with each other is through the hierarchy; no other physical connection exists among them. We introduce the following notation. We use the term LCA (I; j) to denote the least common ancestor of nodes I and j. A parameter that acts the performance of most location management schemes is the relative frequency of move and call operations of each user. This is captured by the call to mobility ratio (CMR). Let Ci be the expected number of calls to user Pi over a time period T and Ui the number of moves made by Pi over T, then CMRi = Ci=Ui. Another important parameter is the local call to mobility ratio LCMRi;j that also involves the origin of the calls. Let Ci;j be the expected number of calls made from zone j to a user Pi over a time period T, then the local call to mobility ratio LCMRi;j is denned as LCMRi;j = Ci;j=Ui. For hierarchical location schemes, the local call to mobility ratio (LCMRi; j) for an internal node j is extended as follows: LCMRi;j = Pk LCMRi;k, where k is a child of j. That is, the local call to mobility ratio for a user Pi and an internal node j is the ratio of the number of calls to Pi originated from any zone at j's sub tree to the number of moves made by Pi. The type of location information maintained in the location databases abets the relative cost of updates and lookups as well as the load distribution among the links and nodes of the hierarchy. Let's

consider rest the case of keeping at all internal databases pointers to lower level databases.

For example, in Figure 3.3(left) for a user x residing at node (cell) 18, there is an entry in the database at node 0 pointing to the entry for x in the database at node 2. The entry for x in the database at node 2 points to the entry for x in the database at node 6, which in turns points to the entry for x in the database at node 18. When user x moves from zone i to zone j, the entries for x in the databases along the path from j to LCA (I; j), and from LCA (I; j) to I are updated. For instance, when user x moves from 18 to 20, the entries at nodes 20, 7, 2, 6, and 18 are updated. Specially, the entry for x is deleted from the databases at nodes 18 and 6, the entry for x at the database at 2 is updated, and entries for x are added to the databases at nodes 7 and 20. When a caller located at zone i places a call for a user x located at zone j, the lookup procedure queries databases starting from node I and proceeding upwards the tree until the rest entry for x is encountered.

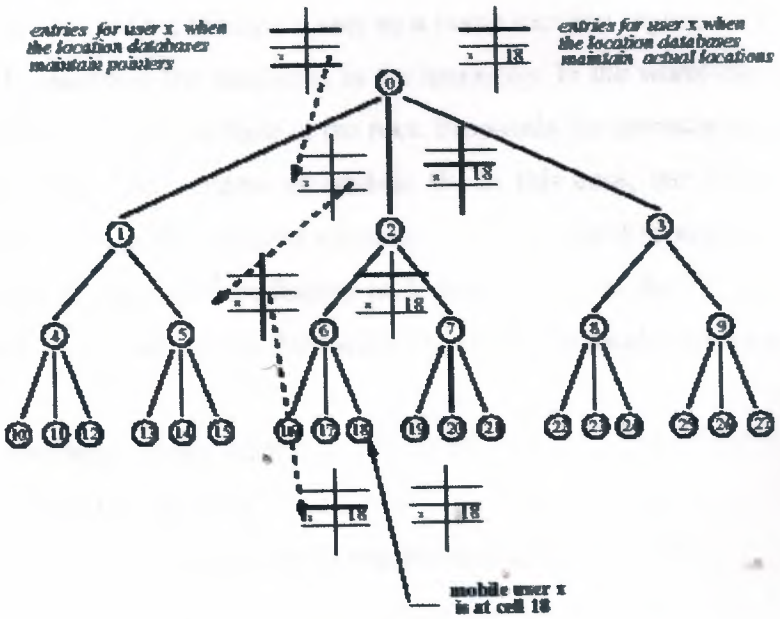

Figure 3.3: Hierarchical Location Schema. Location databases' entries at the left are pointers at lower level databases, while location databases' entries at the right are actual locations.

This happens at node LCA (I; j) (the least common ancestor of nodes I and j). Then, the lookup procedure proceeds downwards following the pointers to node j. For instance, a

call placed from zone 21 to user x located at node 18 (Figure 3.3(left)), queries databases at nodes 21, 7 and nods the rest entry for x at node 2. Then, it follows the pointers to nodes 6 and 18. Let's now consider the case of database entries maintaining the actual location of each user. Then, for user x registered at 18 (Figure 3.3(right)), there are entries in the databases at nodes 0, 2, 6, and 18, each containing a pointer to location 18. In this case, a move from zone I to j cause the update of all entries along the paths from j to the root and from the root to I. For example, a relocation of user x from node 18 to node 20, involves the entries for x at 20, 7, 0, 2, 6, and 18. After the update, entries for x exist in the databases located at nodes 0, 2, 7, and 20, each containing a pointer to 20, while the entries for x in the databases at nodes 6 and 18 were deleted. On the other hand, the cost of a call from i to j is reduced, since once the LCA (I; j) is reached, there is no need to query the databases on the downward path to j. For example, a call placed from node 21 to user x (Figure 3.3(right)), queries databases at nodes 21, 7, 2, and then 18 directly (without querying the database at node 6). When hierarchical location databases are used, there is no need for binding a user to a home location register (HLR). The user can be located by querying the databases in the hierarchy. In the worst case, an entry for the user will be found in the database at the root. Proposals for hierarchical versions have also been made within the context of Mobile IP. In this case, the foreign agents are arranged hierarchically in the regional topology. Each ancestral foreign agent considers the mobile node to be register at the foreign node just below it in the hierarchy.

Table 3.1 Summary of the Pros and Cons of Hierarchical Architectures

(+)	No Pre assigned HLR
(+)	Support for locality
(-)	Increased load and storage requirement at higher level

A hierarchical arrangement of location entries is also possible in ATM networks. A hybrid scheme utilizing both hierarchical entries and pre-assigned home location registers (HLRs) is also possible. Assume that database entries are maintained only at selective nodes of the hierarchy and that an HLR is used. In this case, a call originating from zone i starts searching for the called from zone I. It proceeds following the path from i to the

3.2 Location Management

In mobile computing, mobile objects, e.g., mobile software or users using wireless hardware, may relocate themselves from one network location to another. To enable the ancient tracking of mobile objects, information about their current location may be stored at septic network sites. In abstract terms, location management involves two basic operations, lookups and updates. A lookup or search is invoked each time there is a need to locate a mobile object, e.g., to contact a mobile user or invoke mobile software. Updates of the stored location of a mobile object are initiated when the object moves to a new network location. In the rest of this section, we rest provide an overview of the problem and then introduce network architectures that are commonly associated with mobile computing.

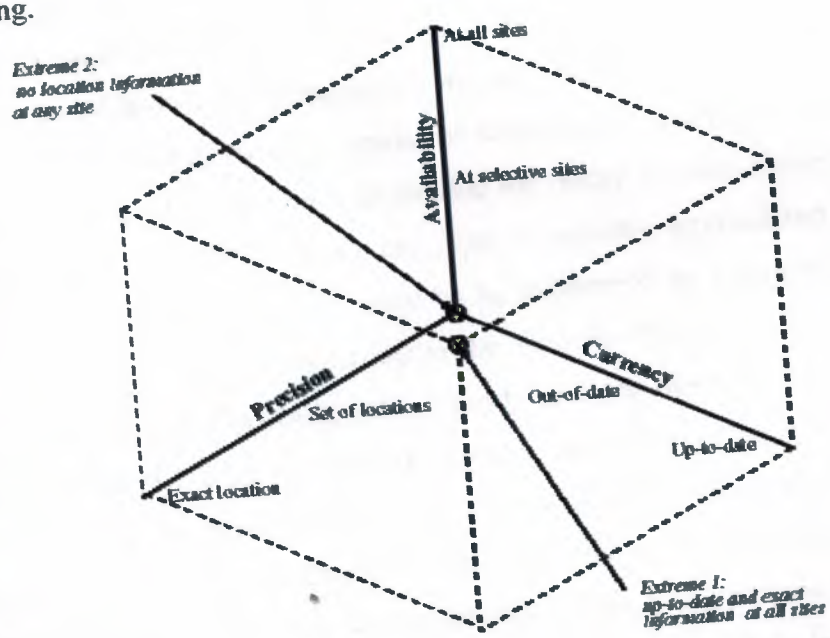

Figure 3.1 Approaches to Saving Location Information

3.2.1 Underlying Network Architecture

The networking infrastructure for providing ubiquitous wireless communication coverage is riper- scented by the personal communication system (PCS) also known by a number of deferent names such as personal communication network (PCN) and UMTS (universal

mobile communication system). While the architecture of the PCS has not evolved yet, it is expected that it will be partially based on the existing digital cellular architecture (see Figure3.2). This network conjuration consists of axed backbone networks extended with a number of mobile hosts (MHz) communicating directly with stationary transceivers called mobile support stations (MSS) or base stations. The area covered by an individual transceiver's signal is called a cell. The mobile host can communicate with other units, mobile or axed, only through the base station at the cell in which it resides. Thus to communicate with a mobile user, the base station of the cell in which it currently resides must be located. As a mobile host moves, it may cross the boundary of a cell, and enter an area covered by a deferent base station. This process is called hand and may involve updating any stored location information for the mobile host. It is speculated that ubiquitous communications will be provided by PCS in a hybrid fashion: heavily populated areas will be covered by cheap base stations of small radius (Pico cells); less populated areas will be covered by base stations of larger radius; and farm land, remote areas and highways with satellites that will provide the bridge between these deferent islands of population density. PCSs involve two types of mobility: terminal and personal mobility Terminal mobility allows a terminal to be indentured by a unique terminal denier independent of its point of attachment to the network. Personal mobility allows PCS users to make and receive calls independently of both their network point of attachment and a septic PCS terminal. Each mobile user explicitly registers itself to notify the system of its current location. The granularity of a registration area ranges from that of a single cell to a group of cells. Once the registration area is indentured, the user can be tracked inside this area using some form of paging. Paging is the process whereby to locate a mobile user, the system issues polling signals in a number of likely locations. By changing the size of a registration area, the edibility of any combination of registration and paging is attained If not explicitly stated otherwise, we use the term cell or zone as synonyms with registration area to indicate a uniquely deniable location where a mobile user can be found. In the cellular architecture, three levels are involved: the access, the axed, and the intelligent network the axed network is the wired backbone network. The access network is the interface between the mobile user and the axed network. The

intelligent network is the network connecting any location registers, i.e., registers used to store information about the location of mobile users.

This network is used to carry track related to tracking mobile users. The Signaling System and its signaling network is a good candidate to carry the signaling track in the intelligent network. Location management is handled at the data link or networking layer transparently from the layers above it each time a call is placed or a change in the network point of attachment occurs. Location management is an issue presents at all wireless networks (e.g., cellular, wireless LANs, and satellites). Although most solutions so far relate to cellular architectures at the data link Layer and to wireless LAN architectures at the networking layer, most are general enough to be applicable to deferent layers and architectures. In addition to handling mobility at lower layers, the need for information about the location of moving objects is encountered at the application level as well.

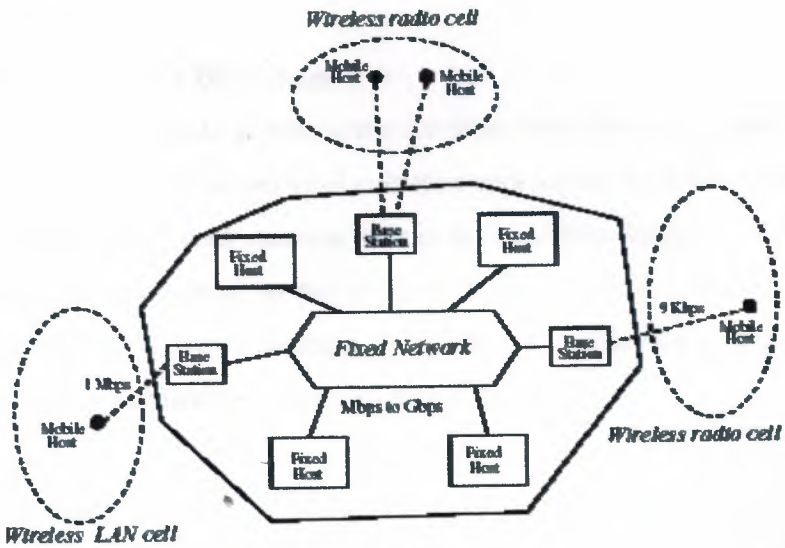

Figure 3.2 Wireless Computing Architecture

Applications may need information about the location of mobile users to answer a Variety of queries that involve location Other applications may involve updating environmental parameters and selecting locally available computing resources (e.g., nearest printer) There is no standard way for applications to acquire and use location

information. For example, applications may choose to maintain their own data structures for storing location information. The cellular architecture is not the sole infrastructure for wireless mobile computing. In its absence, various techniques may be employed to identify the current location of mobile users, for instance, users may be equipped with a Global Positioning System (GPS).GPS are space-based radio positioning systems that provide three-dimensional position, velocity and time information to suitably equipped users anywhere on or near the surface of the Earth. Common applications in this area include digital battle ends in the military context and transportation systems in the civilian industry. Finally, besides mobility tied to wireless hardware, the techniques presented in this paper are also applicable when the objective is to locate mobile code and data. Furthermore, similar techniques are also necessary when instead of location; the objective is to anciently retrieve other parole information related to mobile users. This information may include Quos related parameters or services.

3.3 Architectures of Location Databases

In this section, we describe basic architectures for distributed databases used for storing the location of moving users. The two most common approaches are a two-tier scheme in which the current location of each moving user is saved at two network locations and a tree-structured distributed database in which space is hierarchically decomposed in sub regions. We also describe a graph theoretic approach that employs regional directories. Finally, we refer brie to a centralized database approach.

3.3.1 Two-tier Schemes

In two-tier schemes, a home database, termed Home Location Register (HLR), is associated with each mobile user. The HLR is located at network location (zone) pre specie for each user. It maintains the current location of the user as part of the user's parole. The search and update Procedures are quite simple. To locate a user x, ax's HLR is indentured and queried. When a user x moves to a new zone, ax's HLR is contacted and updated to maintain the new location. As an enhancement to the above scheme, Visitor Location Registers (VLSI) is maintained at each zone. The VLR at a zone stores copies of paroles of users not at their home location and currently located inside that

zone. When a call is placed from zone I to user x, the VLR at zone I is queried rest and only if the user is not found there, is ax's HLR contacted. When a user x moves from zone I to j, in addition to updating ax's HLR, the entry for x is deleted from the VLR at zone I, and a new entry for x is added to the VLR at zone j. The two prevailing existing standards for cellular technologies, the Electronics Industry Association Telecommunications Industry Associations (EIA/TIA) Interim Standard commonly used in North America and the Global System for Mobile Communications (GSM) used in Europe, both support carrying out location strategies using Hers and VLSI at the Internet networking level, mobile IP is a medication to wire line IP that allows users to continue to receive messages independently of their point of attachment to the Internet. Mobile IP is designed within the IETF (Internet Engineering Task Force) and is outlined in a number of Requests for Comments (RFC).Wire line IP assumes that the network address of a node uniquely indentures the node's point of attachment to the Internet. Thus, a node must be located on the network indicated by its IP address to receive messages destined to it. To remedy this, in mobile IP, there are two IP addresses associated with each mobile node. One address, known as the home address of the node, is used to identify the node and is treated administratively just like a permanent IP address. When away from its home network, a care-of-address is associated with the mobile node and re acts the mobile node's current point of attachment. The care-of-address is either the address of a foreign agent which is a router on the visited network that provides services to the mobile node or a co-located address which is an address temporarily acquired by the mobile node. When a mobile node is away of its home, it registers its care-of-address with its home address. Then to deliver any messages, the home agent tunnels them to the care-of-address. One problem with the home location approach is that the assignment of the home register to a mobile object is permanent. Thus, long-lived objects cannot be appropriately handled, since their home location remains axed even when the objects permanently move to a deferent region. Another Drawback of the two-tier approach is that it does not scale well with highly distributed systems where sites are geographically widely dispersed. To contact an object, the possibly distant home location must be contacted rest. Similarly, even a move to a nearby location must be registered at a

potentially distant home location. Thus, locality of moves and calls is not taken advantage off.

3.3.2 Hierarchical Schemes

Hierarchical location schemes extend two-tier schemes by maintaining a hierarchy of location databases. In this hierarchy, a location database at a higher level contains location information for users located at levels below it. Usually, the hierarchy is tree-structured. In this case, the location database at a leaf serves a single zone (cell) and contains entries for all users registered in this zone.

A database at an internal node maintains information about users registered in the set of zones in its sub tree. For each mobile user, this information is either a pointer to an entry at a lower level database or the user's actual current location. The databases are usually interconnected by the links of the intelligent signaling network, e.g., a Common Channel Signaling (CCS) network. For instance, in telephony, the databases may be placed at the telephone switches. It is often the case that the only way that two zones can communicate with each other is through the hierarchy; no other physical connection exists among them. We introduce the following notation. We use the term LCA (I; j) to denote the least common ancestor of nodes I and j. A parameter that acts the performance of most location management schemes is the relative frequency of move and call operations of each user. This is captured by the call to mobility ratio (CMR). Let Ci be the expected number of calls to user Pi over a time period T and Ui the number of moves made by Pi over T, then CMRi = Ci=Ui. Another important parameter is the local call to mobility ratio LCMRi;j that also involves the origin of the calls. Let Ci;j be the expected number of calls made from zone j to a user Pi over a time period T, then the local call to mobility ratio LCMRi;j is denned as LCMRi;j = Ci;j=Ui. For hierarchical location schemes, the local call to mobility ratio (LCMRi; j) for an internal node j is extended as follows: LCMRi;j = Pk LCMRi;k, where k is a child of j. That is, the local call to mobility ratio for a user Pi and an internal node j is the ratio of the number of calls to Pi originated from any zone at j's sub tree to the number of moves made by Pi. The type of location information maintained in the location databases abets the relative cost of updates and lookups as well as the load distribution among the links and nodes of the hierarchy. Let's

consider rest the case of keeping at all internal databases pointers to lower level databases.

For example, in Figure 3.3(left) for a user x residing at node (cell) 18, there is an entry in the database at node 0 pointing to the entry for x in the database at node 2. The entry for x in the database at node 2 points to the entry for x in the database at node 6, which in turns points to the entry for x in the database at node 18. When user x moves from zone i to zone j, the entries for x in the databases along the path from j to LCA (I; j), and from LCA (I; j) to I are updated. For instance, when user x moves from 18 to 20, the entries at nodes 20, 7, 2, 6, and 18 are updated. Specially, the entry for x is deleted from the databases at nodes 18 and 6, the entry for x at the database at 2 is updated, and entries for x are added to the databases at nodes 7 and 20. When a caller located at zone i places a call for a user x located at zone j, the lookup procedure queries databases starting from node I and proceeding upwards the tree until the rest entry for x is encountered.

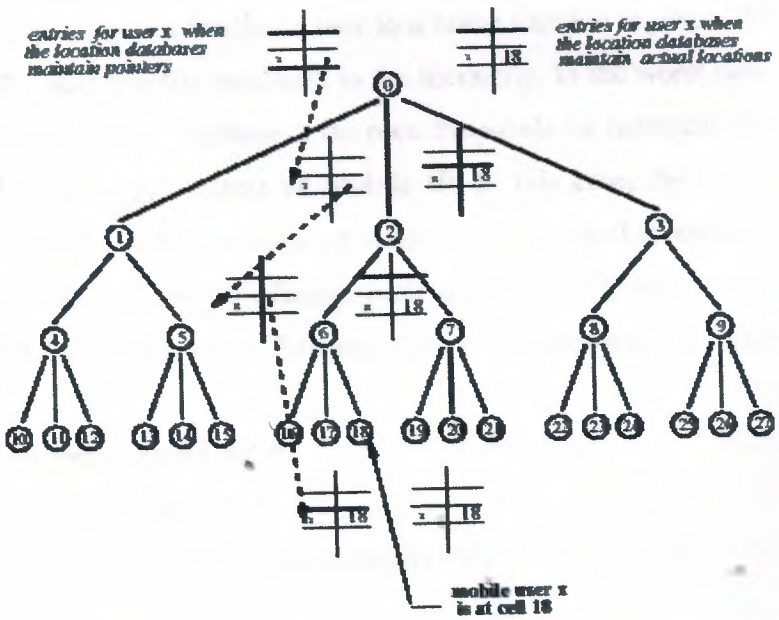

Figure 3.3: Hierarchical Location Schema. Location databases' entries at the left are pointers at lower level databases, while location databases' entries at the right are actual locations.

This happens at node LCA (I; j) (the least common ancestor of nodes I and j). Then, the lookup procedure proceeds downwards following the pointers to node j. For instance, a

call placed from zone 21 to user x located at node 18 (Figure 3.3(left)), queries databases at nodes 21, 7 and nods the rest entry for x at node 2. Then, it follows the pointers to nodes 6 and 18. Let's now consider the case of database entries maintaining the actual location of each user. Then, for user x registered at 18 (Figure 3.3(right)), there are entries in the databases at nodes 0, 2, 6, and 18, each containing a pointer to location 18. In this case, a move from zone I to j cause the update of all entries along the paths from j to the root and from the root to I. For example, a relocation of user x from node 18 to node 20, involves the entries for x at 20, 7, 0, 2, 6, and 18. After the update, entries for x exist in the databases located at nodes 0, 2, 7, and 20, each containing a pointer to 20, while the entries for x in the databases at nodes 6 and 18 were deleted. On the other hand, the cost of a call from i to j is reduced, since once the LCA (I; j) is reached, there is no need to query the databases on the downward path to j. For example, a call placed from node 21 to user x (Figure 3.3(right)), queries databases at nodes 21, 7, 2, and then 18 directly (without querying the database at node 6). When hierarchical location databases are used, there is no need for binding a user to a home location register (HLR). The user can be located by querying the databases in the hierarchy. In the worst case, an entry for the user will be found in the database at the root. Proposals for hierarchical versions have also been made within the context of Mobile IP. In this case, the foreign agents are arranged hierarchically in the regional topology. Each ancestral foreign agent considers the mobile node to be register at the foreign node just below it in the hierarchy.

Table 3.1 Summary of the Pros and Cons of Hierarchical Architectures

(+)	No Pre assigned HLR
(+)	Support for locality
(-)	Increased load and storage requirement at higher level

A hierarchical arrangement of location entries is also possible in ATM networks. A hybrid scheme utilizing both hierarchical entries and pre-assigned home location registers (HLRs) is also possible. Assume that database entries are maintained only at selective nodes of the hierarchy and that an HLR is used. In this case, a call originating from zone i starts searching for the called from zone I. It proceeds following the path from i to the

LCA of I and the caller's HLR and then moves downwards to the caller's HLR, unless an entry for the called is found in any database on this path. If such an entry is encountered, it is followed instead. The hierarchical scheme leads to reductions in communication cost when most calls and moves are geographically localized. In such cases, instead of contacting the HLR of the user that may be located far away from the user's current location, a small number of location databases in the user's neighborhood are accessed. However, the number of location databases that are updated and queried increases relative to the two-tier scheme. Another problem with the hierarchical schemes is that the databases located at higher-level must handle a relatively large number of messages. Furthermore, they have large storage demands. One solution is to partition the databases at the high-level nodes (e.g., at the root) into smaller databases at sub-nodes so that the entries of the original database are shared appropriately among the databases at the sub-nodes. Table3.1 summarizes some of the pros and cons of the hierarchical architectures.

3.3.3 Non-tree Hierarchy: Regional Matching

The objective of the regional directories approach is to favor local operations, in that moves to nearby locations or searches for nearby users cost less. The approach guarantees communication overheads that are polylogarithmic in the size (i.e., number of network sites) and the diameter (i.e., maximum distance between any two sites) of the network. The overhead is evaluated by comparing the total cost of a sequence of move and call operations against the inherent cost, i.e., the cost incurred by the operations assuming that information for the current location of each user exists at all sites for free. The comparison is done over all possible sequences of move and call operations. Location databases called regional directories are organized in a non-tree hierarchy. In particular, a hierarchy D of regional directories is built, where = log d, for d being the maximal distance between any two network sites. The purpose of a regional directory RDi at level i am to enable a potential searcher to track any user residing within distance 2i from it. Two sets of sites are associated with each site u in an RDi directory: a read set Ready(u) and a write set Write(u) with the property that the read set Ready(u) and the write set Write(w) intersect for any pair of site u and w within a distance 2i from each other. The two sets of sites are used as follows. Each site reports all users it hosts to every site in its

write set and upon looking for a user, it queries all sites in its read set. Whenever a user moves to a new location at distance k away, only the log k lowest levels of the hierarchy are updated to point directly to the new address. Directory entries at higher level directories continue pointing to the old address, where a forwarding pointer to the new location is left. To bound the length of the chain of forwarding pointers, it is guaranteed that for every user the distance C(x) traveled since its address was updated at the regional directory RDi is less or equal to 2i_1 _ 1 for each level I. The complete search and update procedures follow.

Regional Matching Search Procedure

/* a call is placed from a user at site w to user x */

I 0 address nil

Repeat

I I + 1

/* Search directory RDi */

For all sites u in Read (w)

Query u

Until address <> nil

Repeat

Follow forwarding pointers

Until reaching x

Regional Matching Move Procedure

/* user x moves from site v to site w */

Let RDj be the highest directory for which C(x) > 2j_1 _ 1

For I =1 to maxfJ; _g

/* Update directory RDi */

For all sites u in Write (v)

Update entry

Add a forwarding pointer at RDi+1

3.3.4 A Centralized DBMS

The architectural alternatives presented so far are distributed, in that the locations of moving objects are stored in deferent network sites. For some applications, it is feasible to use a centralized Approach in which the locations of all moving objects are stored in a single centralized Database Management System (DBMS). Such applications include for example a trucking company's database, a database representing the location of taxi-cabs or in the context of military applications, a database that keeps track of the position of all moving objects in a battened. In this case, all location queries and updates are directed to the central DBMS. Using an existing spatial DBMS is not scent, since existing DBMS do not handle well continuously changing data, such as the location of moving objects. Thus, most current research in this area deals with extending spatial databases with such capabilities.

3.4 Placement of Databases

Maintaining location information at all nodes in the hierarchy results in cost-elective lookups. However, it increases the number of databases that must be updated during each move operation. To reduce the update cost, database entries may be only selectively maintained at specie nodes in the tree hierarchy. In this case, during the search and update procedures, only nodes that contain location databases are queried or updated; others are skipped. For instance, when a call is made from j to I the search procedure traverses the tree from node j up to the lowest level ancestor of the LCA (I; j) that contains a location database. A possible placement of location databases is to maintain location entries for mobile hosts only at the leaf nodes of the zone in which they reside currently. In this case, when there is no home location register associated with a mobile host, some form of global searching in the hierarchy is needed to nod its current location. In this scenario, location strategies include at, expanding, and hybrid searches. Let home be the zone at which a user registers initially. The at search procedure starts from the root, and then in turn queries in parallel all nodes at the next level of the tree until the leaf level is reached. The expanding search procedure starts by querying the home of the called I, then queries the parent of the home, which in turn queries all its children and so on. This type of search favors moves to nearby locations. Finally, the hybrid search

procedure starts as the expanding one, but if the location is not found at the children of the parent of the caller's home, at search is initiated. The hybrid scheme can locate quickly those users that when not at home happen to be found far away from it. Next we consider three alternative architectures: maintaining location information at selective internal nodes so that some performance metric is optimized, dynamic hierarchical database architecture, and partitions.

3.4.1 Optimization

The placement of location databases in the hierarchy can be seen as an optimization problem. Objective functions include minimizing: (a) the number of database updates and accesses, (b) the communication cost, (c) the sum of the track on the network link or links, or any combination of the above. Constraints that must be stained include: (a) an upper bound on the rate at which each database can be updated or accessed, (b) the capacity of links, and (c) the available storage. Such an optimization-based approach is taken in. The objective there is to minimize the number of updates and accesses per unit of time given a maximum database service capacity (i.e., the maximum rate of updates and lookups that each database can service) and estimates of the call to mobility ratio. In this approach, communication is not considered, and thus, if the service capacity is saliently large, a single, central database at the root is the optimal placement. The problem is formulated as a combinatorial optimization problem and is solved using a dynamic programming algorithm.

3.4.2 Dynamic Hierarchical Database Architecture

The dynamic hierarchical scheme proposed in extends the two-tier scheme by introducing a new level of databases called directory registers (DRs). Each DR covers a number of location zones. Its primary function is to periodically compute and store the location conjuration for the mobile units located in zones under its service. There are three types of location addresses that can be stored at a DR. In particular, besides maintaining the local address of all mobile units located in its coverage, each DR also maintains for selected mobile units either a direct remote address to their current location or an indirect remote address to their current serving DR. For each particular mobile unit, the selection

of the set of DRs that maintain direct remote addresses or indirect remote addresses for it is periodically determined based on the mobility and call arrival patterns of the unit. The HLR may either store the current zone or the current DR of a mobile unit, again depending on the mobility or call arrival patterns of the unit. In the cases that it is more cost elective not to set up any remote addresses, the scheme reduces to the original two-tier scheme. In contrast to the two-tier and the hierarchical architectures where the strategy for the distribution of location information is the same for all mobile units, in this scheme, the distribution strategy is dynamically adjusted for each mobile unit.

3.4.3 Partitions

To avoid maintaining location entries at all levels of the hierarchy, and at the same time reduce the search cost, partitions are deployed. The partitions for each user are obtained by grouping the zones (cells) among which it moves frequently and separating the zones between which it relocates infrequently. Thus, partitions exploit locality of movement. Partitions can be used in many ways. We describe next two such partition-based strategies.

For each partition, the information whether the user is currently in the partition is maintained at the least common ancestor of all nodes in the partition, called the representative of the partition. The representative knows that a user is in its partition but not its exact location. This information is used during at search (i.e., top-down search starting from the root) to decide which sub tree in the hierarchy to search. Thus, partitions reduce the overall search cost as compared to at search. There is an increase however on the update cost since, when a user crosses a partition; the representatives of its previous and new partitions must be informed. For example, assume that user x often moves inside four deferent set of nodes, i.e., partitions, and infrequently between these sets. The nodes of each partition are f10, 12, 14, 15g, f16, 18g, f19, 20,21g and f22, 23, 25, 26, 27g and are depicted in Figure 3.4. The representative node of each partition is high-lighted. When user x is at node 14 in partition 1, the representative of the associated partition, node 1, maintains the information that the user is inside its partition. When user x moves to node 16 that is outside the current partition, both node 1, the representative of the old

partition, and node 6, the representative of the new partition, are updated to react the movement.

A slightly deferent use of partitions called redirection trees is proposed in. A single partition, called local region, is denned by including all nodes between which the user often moves.

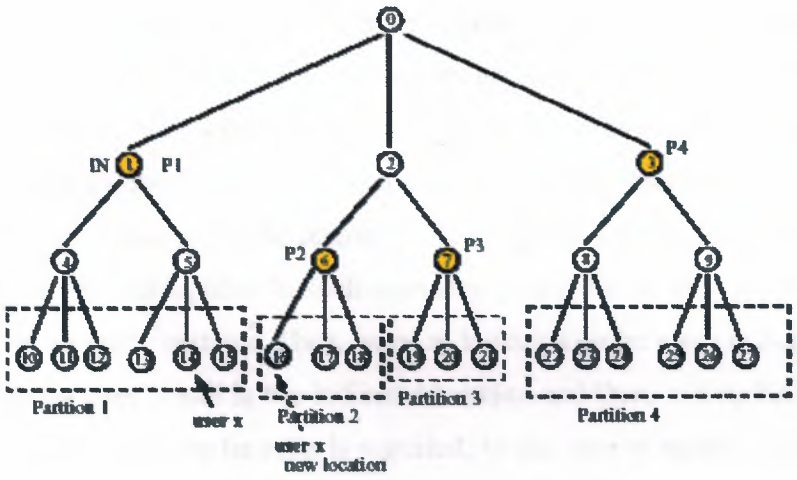

Figure 3.4: Partitions

The representative of the local region called a redirection agent maintains the location of all users that have appointed it as their redirection agent. When the user is located in its local region, its redirection agent redirects any calls passing through it during any type of search (e.g., at or using HLRs) to the current location of the user. Movements inside a local region are recorded in the redirection agent and not necessarily at location servers outside the region.

3.5 Caching

Caching is based on the premise that after a call is resolved, the information about the current location of the called should be reused by any subsequent calls originated from the same region. To this end, in two-tier architectures, every time a user x is called; ax's location is cached at the VLR in the caller's zone, so that any subsequent call to x

originated from that zone can reuse this information. Caching is useful for those users who receive calls frequently relative to the rate at which they relocate. Similar to the idea of exploiting locality of le accesses, the method exploits the spatial and temporal locality of calls received by users.

To locate a user, the cache at the VLR of the caller's zone is queried rest. If the location of the user is found at the cache, then a query is launched to the indicated location without contacting the user's HLR. Otherwise, the HLR is queried.

Regarding cache invalidation, there are various approaches. In eager caching, every time a user moves to a new location, all cache entries for this user's location are updated. Thus, the cost of move operations increases for those users whose address is cached. In this type of caching, the locations of the cache entries for a user's location must be centrally known in order for the updates to be initiated. This leads to scalability problems as well as making the scheme susceptible to fault tolerance problems. In lazy caching, a move operation signals no cache updates. Then, when at lookup a cache entry is found there are two cases: either the user is still in the indicated location and there is a cache hit, or it has moved out, in which case a cache miss is signaled. In the case of apache miss, the usual procedure is followed: the HLR is contacted and after the call is resolved the cache entry is updated. Thus, in lazy caching, the cached location for any given user is updated only upon a miss. The basic overhead involved in lazy caching is in cases of cache misses, since the cached location must be visited rest. So, for lazy caching to produce savings over the non-caching scheme, the hit ratio p for any given user at a specie zone must exceed a hit ratio threshold pT = CH=CB, where CH is the cost of a lookup when there is a hit and CB the cost of the lookup in the non-caching scheme. Among other factors, CH and CB depend on the relative cost of querying HLR's and V LR's. A performance study for lazy caching is presented in. There, an estimation of CH and CB is computed for a given signaling architecture based on a Common Channel Signaling network that uses the SS7 protocol to set up calls. Conclusions are drawn on the bents of caching based on which of the factors participating in CH and CB dominate. The hit ratio for the cache of user's i location at zone j can also be directly related to the LCMRi;j of the user . For instance, when the incoming calls follow a Poisson distribution with arrival rate and the interwove times are exponentially distributed with mean _, then p = _= (_+_) and the

minimum LCMR, denoted LCMRT, required for caching to be Benicia is LCMRT = pT = (1 _ pT). So, caching can be selectively done per user I at zone j, when the LCMi;j is larger than the LCMRT bound. In general, this threshold is lower when users accept calls more frequently from users located nearby. In practice, it is expected that LCMRT > 7. Another approach to cache invalidation, suggested in, is to consider cache entries obsolete after a certain time period. To determine when a particular cache should be cleared, a threshold T is used. T it dynamically adapted to the current call and mobility patterns such that the overall network track is reduced. When the cache size is limited, cache replacement policies, such as replacing the least recently used (LRU) location, may be used. Another issue is how to initialize the cache entries. User paroles and other types of domain knowledge may be used to initially populate the cache with the locations of the users most likely to be called. In mobile IP, route optimization provides a means for any node to maintain a binding cache containing the care-of-address of one or more mobile nodes. Such cache entries are used by the sender to tunnel any messages directly to the care-of-address indicated in its cache. Each entry in the binding cache has an associated lifetime that is specie when the entry is created. The entry is to be deleted from the cache after the expiration of this time period. A lazy procedure is also used to update out-of-date cache entries.

In the approach we have described, caching is performed on a per-user basis: the cache maintains the address of the last called users. Another approach is to apply a static form of caching, e.g., by caching the addresses of a certain group of users or certain parts of the network where the users' call to mobility ratios (CMRs) are known to be high on average.

Caching techniques can also be deployed to exploit locality of calls in tree-structured hierarchical architectures. Recall that in hierarchical architectures, when a call is placed from zone I to user x located at zone j, the search procedure traverses the tree upwards from I to LCA (I; j) and then downwards to j.

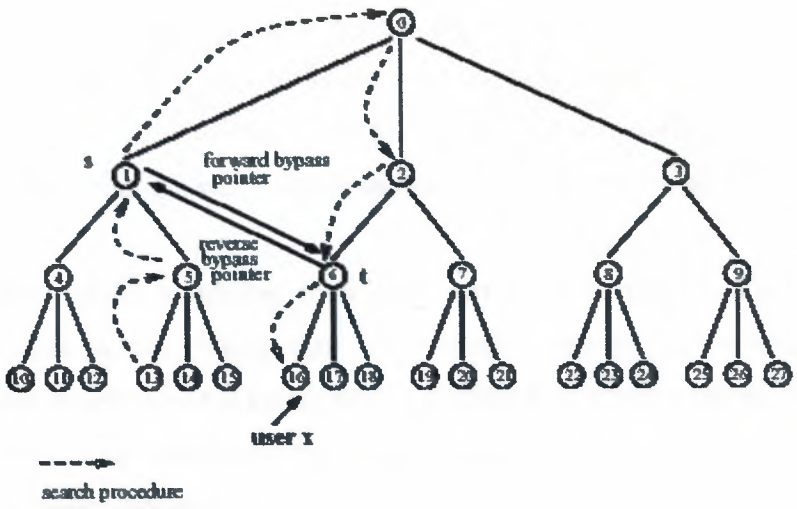

Figure 3.5: Caching in Hierarchical Location Schemes. For simplicity, the acknowledgment message is not shown; it follows the reverse route of the search procedure.

We also consider an acknowledgment message that returns from j to I. To support caching, during the return path, a pair of bypass pointers, called forward and reverse, is created. A forward bypass pointer is an entry at an ancestor of I, say s that points to an ancestor of j say t; the reverse bypass pointer is from t to s. During the next call from zone i to user x, the search message traverse the tree upwards until s is reached. Then, the message travels to database either via LCA (I; j) or via a shorter route if such a route is available in the underlying network. Similarly, the acknowledgment message can bypass all intermediate pointers on the path from t to s. For example, let a call be placed from zone 13 to user x at zone 16 (Figure3.5). A forward bypass Pointer is set at node 1 pointing to node 6; the reverse bypass pointer is from 6 to 1. During the next call from zone 13 to user x, the search message traverses the tree from node 13 up to node 1 and then at node 6, either through LCA (1; 6), that is node 0, or via a shorter path. In any case, no queries are posed to databases at nodes 0 and 2. The level of nodes s and t where the bypass pointers are set varies. In simple caching, s and t are both leaf nodes, while in level caching, s and t are nodes belonging to any level and possibly each to a deferent one (as in the previous example). Placing a bypass pointer at a high-level node s, makes this

entry available to all calls originated from zones at sis sub tree. However, calls must traverse a longer path to reach s. Placing the pointer to point to a high-level node t, increases the cost of lookup, since to locate a user, a longer path from t to the leaf node must be followed. On the other hand, the cache entry remains valid as long as the user moves inside its sub tree. An adaptive scheme can be considered to set the levels of s and t dynamically.

As in the two-tier location scheme, there are many possible variations for performing cache invalidations. In lazy caching, the move operation remains unchanged, since cache entries are updated only when a cache miss is signaled. In eager caching, cache entries are updated at each move operation. Specially, consider a move operation from zone i to zone j, where a registration/deregistration message propagates from j via LCA (j; i) to i. During this procedure, the bypass pointers which are no longer valid are deleted. These pointers include any forward bypass pointers found during the upward traversal of the registration message, and any reverse or bypass pointers found during the downward traversal of the deregistration message. Preliminary performance results are reported in. The analysis is based on a quantity called Regional Call-to-Mobility Ratio (RCMR) denned for a user x with respect to tree nodes s and t as the average number of calls from the sub tree rooted at s to user x, while user x is in the sub tree rooted at t. It is shown, that under certain assumptions, for users with RCMR > 5, caching can result in up to a 30% reduction in the cost of both calls and moves, when considering only the number of database operations. Caching in the case of storing the exact location at internal nodes, as opposed to pointers to lower level databases, can also be deployed in many ways again ranging from simple to level caching. In simple caching, the current location of the user is cached only at leaf nodes. In level caching, the current location of a user is cached at all nodes up to a given level. Caching is orthogonal to partitions. In fact, in caching is used in conjunction with partitions. In particular, instead of caching the current location of the called, the location of its representative is cached. For example, assume that partitions are denned as in Figure 4 and user x is at node 14. Let a call be placed for user x. Instead of caching location 14 (or a pointer to it), location 1, e.g., the representative of the current partition, is cached. This ignorantly reduces the cost of cache updates, since a cache entry becomes obsolete only when a user moves outside the Current partition.

3.6 Replication

To reduce the lookup cost, the location of specie users may be replicated at selected sites. Reply action reduces the lookup cost, since it increases the probability of ending the location of the called locally as opposed to issuing a high latency remote lookup. On the other hand, the update cost incurred increases considerably, since replicas must be maintained consistent every time the user moves. In general, the location of a user i should be replicated at a zone j, only if the replication is judicious, that is the savings due to replication exceed the update cost incurred. As in the case of caching, the beets depend on the LCMR. Intuitively, if many calls to I originate from zone j, then it makes sense to replicate I at j. However, if I move frequently, then replica updates incur excessive costs. Let be the cost savings when a local lookup, i.e., a query of the local VLR, succeeds as opposed to a remote query and the cost of updating a replica, then a replication of the location of user I at zone j is judicious if where Ci; j is the expected number of calls made from zone j to I over a time period T and Ui the number of moves made by I over T.

In addition to cost, the assignment of replicas to zones must take into account other parameters, such as the service capacity of each database and the maximum memory available for storing replicas. The replication sites for each user may be kept at its HLR. Besides location information, other information associated with mobile users may also be replicated. Such information may include service information such as call blocking and call forwarding, as well as Quos requirements such as minimum channel quality or acceptable bandwidth. Unlike location information which is needed at the caller's region, service and Quos information is needed at the location at which the call is received. Approaches similar to those used for replication of location information can be used to replicate service information at sites that are frequently visited by a mobile user in place of sites from which most calls for that user originate. Finally, instead of the exact location of a user, more coarse location information, e.g., the user's current partition may be replicated. The coarseness or granularity of location replicas presents location schemes with a trade between the update and the lookup costs. If the information replicated is coarse then it needs to be updated less frequently in the expense of a higher lookup resolution cost. Choosing the network sites at which to maintain replicas of the current

location of a mobile user resembles the allocation and the database allocation problem. These classical problems are concerned with the selection of sites at which to maintain replicas of database partitions.

The selection of sites is based on the read/write pattern of each partition, that is the number of read and write operations issued by each site. In the case of location management, this corresponds to the lookup/update pattern of a user's locations. Most schemes for the database allocation are static that is they are based on the assumption that the read/write pattern does not change. We describe four per-user replication schemes. The rest one takes into account resource restrictions and is centralized, whereas the second one does not place any such global restrictions and thus is distributed. The rest two algorithms are for two-tier schemes, while the third one is applicable to tree-structured hierarchical architectures. The last algorithm is not developed specially for location management but treats the problem of dynamic data allocation in its general form. It is a distributed algorithm that considers no global restrictions. It is applicable to any architecture, but it is proven to be optimal for tree-structured hierarchical schemes.

3.6.1 per User Profile Replication

The objective of the per user parole approach is to minimize the total cost of moves and calls, while maintaining constraints on the maximum number ri of replicas per user Pi and on the maximum number pj of replicas stored in the database at zone Zj . Let M be the number of users and N be the number of zones. A replication assignment of a user's parole Pi to a set of zones R (Pi) is found, such that the system cost expressed as the sum: Is minimized and any given constraints on the maximum number of replicas per database at each zone and on the maximum number of replicas per user are maintained. To this end, an own network F is constructed as follows. The vertices of the graph correspond

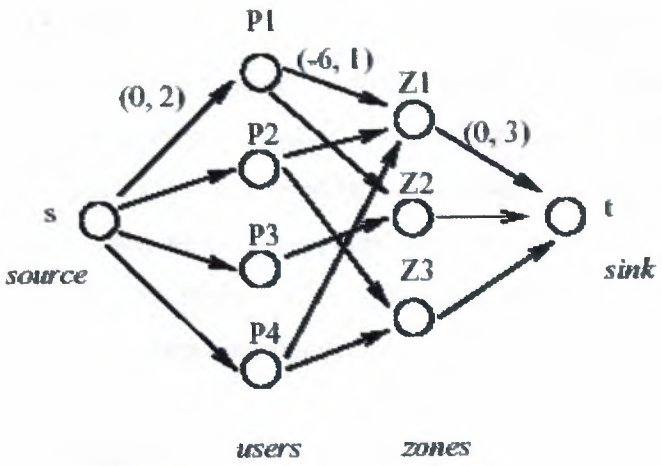

Figure 3.6: Example of a Flow Network

There are two special vertices, a source vertex s, and a sink vertex t. A pair (c; p) of a cost, c, and a capacity, p, attribute is associated with each edge. An edge is added from s to all Pi with (c; p) = (0; RI) and from all Zj to t with (0; pj). An edge from Pi to Zj with Is added only if it is judicious to replicate Pi at Zj, i.e., if Inequality (1) holds. Then, computing a minimum-cost (min-cost) maximum-own (max-own) on F nods the requested assignment. In Figure3.6, a simple own network of a system with four mobile users and 3 zones is depicted. The capacity attribute 2 on edge (s; P1) indicates that P1's parole can be replicated in at most two zones. The capacity attribute 3 on edge (Z1; t) indicates that the database at zone Z1 can store at most three replicas. Finally, in the pair (_6; 1) on edge (Z1; P1), the cost attribute -6 indicates that replicating P1's parole in zone Z1 will yield a net cost saving of six over not replicating, while the capacity attribute 1 indicates that P1 should be replicated at most once in Z1.

In such a centralized approach, generating, distributing and applying to all sites, a particular replica assignment decision is a time consuming, computational intensive, and bandwidth demanding process. Thus, computing and applying a new replication plan is very expensive, and a graceful adaptation of the replica assignment to changing calling and mobility patterns is very important [56]. Let Few represent the own network solution for a new calling and mobility pattern new and Fold represent the own network solution for the previous pattern old. An algorithm is presented that incrementally computes the

min-cost max-own of few given the min-cost max-own of Fold .A desired property of the replica assignment algorithm is to keep the cost of evolution from Fold to Few low by avoiding radical changes in the replication plan. To this end, two approaches are proposed: (1) a tempered min-cost max-own, that factors in the cost of replica reassignments when augmenting paths, and (2) a minimum mean cycle canceling algorithm, that augments ow along cycles with the minimum mean cost, where the cost expresses the number of replica reassignments.

3.6.2 Working Set Replication

The working set method relies on the observation that each user communicates frequently with a small number of sources, called its working set, thus it makes sense to maintain copies of its location at the members of this set. The approach is similar to the per-user replication except from the fact that no constraints are placed on the database storage capacity or the number of replicas per user. Consequently, the decision to provide the information of the location of a mobile unit Pi at a zone Zj can be made independently at each unit Pi. Specially, Inequality (1) is evaluated locally at the mobile unit each time at least one of the quantities involved in the inequality changes. This happen: (a) each time a call is set up and (b) when the mobile unit moves. In the former case, the inequality is evaluated only if the caller's site is not a member of the working set of the called. If the inequality is found to hold, the caller's site becomes a member of the set. In the later case, the inequality is re-evaluated for all members of the working set, and the members for which the inequality no longer holds are dropped the set. This way the scheme adapts to the current call and mobility pattern. Note that in case (a) all four terms of Inequality (1) need to be recomputed, while in case (b) only the number of moves (Ui) needs to be re-evaluated. Simulation studies in show that, as expected, when the call to mobility ratio (CMR) value is low the scheme performs like a scheme without replication. When the CMR value is high, the scheme behaves like a static scheme in which the working set for a user is axed. It is also shown that the performance of this adaptive scheme is not primarily acted by the number of units in the working set but rather by the CMR of each individual unit.

3.6.3 Replication in Hierarchical Architectures

In hierarchical architectures, in addition to leaf nodes, the location of a mobile user may be selectively replicated at internal nodes of the hierarchy. As in the replication schemes for two-tier architectures, the location of a user should be replicated at a node only if the cost of replication does not exceed the cost of non replication. However, in a hierarchical location database scheme, if a high LCMR value is the determining factor for selecting replication sites, then the databases at higher levels will tend to be selected as replication sites over databases at lower levels, since they possess much higher LCMR values. In particular, if a database at level j is selected, all it ancestors are selected as well. Recall that the LCMR for an internal node is the sum of the LCMR s of its children. Such a selection would result in excessive update activities at higher-level databases. To compensate, replication algorithms for hierarchical databases must also set some maximum level of the hierarchy at which to replicate. Hi Per proposed in [33] is a family of location management techniques with four parameters: N max, S min, S max and L, where N max determines the maximum number of replicas per user, S min and S max together determine when a node may be selected as a replication site, and L determines the maximum level of the hierarchy at which replicas can be placed. The location of user I is not replicated at j if LCMRi;j is smaller than Sin, while it is replicated if LCMRi;j exceeds S max. If S min _ LCMRi;j < S max, then whether replication should be performed or not depends on a number of constraints placed by the database topology. The constraints taken into account by Hi per are N max and L. An o_-line algorithm to compute the sites of replication for each user I proceeds in two phases. In the rest phase, in a bottom-up traversal, it allocates replicas of I at all databases with LCMRi; j _ S max as long as the number of allocated replicas n does not exceed N max. In the second phase, if n _ N max, the algorithm allocates the remaining replicas to databases below level L with the largest non negative LCMRi;j _ S max in a top-down fashion. The optimal values S opt min and S opt max for S min and S max are determined based on whether replication is judicious, that is, if the beets of replication exceeds its costs.

3.6.4 The ADR Algorithm

The Adaptive Data Replication (ADR) algorithm presents a solution to the general problem of determining an optimal (in terms of communication cost) set of replication sites for an object in a distributed system, when the object's read-write pattern changes dynamically. We will describe the ADR algorithm for the case of tree-structured architectures. The tree represents a physical or logical communication structure. Two sites are neighbor sites if they are connected through a tree edge. Let R be the current replication set of object x, i.e., the sites at which x is replicated currently. A read of object x is performed from the closest replica in R, while a write of x updates all replicas in R. Metaphorically, the replication set R forms a variable-size amoeba that stays connected at all times and constantly moves towards the center of the read-write activity. The ADR algorithm updates the replication set R of each object x periodically at a time period T. The replication set expands as the read activity increases and contracts as the write activity increases. Specially, at the end of the time period T, specie sites of the network perform three tests, namely the expansion, the contraction and the switch test described below. First, we introduce related terminology. A site R-neighbor, if it belongs to R but has a neighbor site that does not belong to R. If site I is not a singleton set, site I is a Fringe site, if it is a leaf at a sub graph induced by R. The expansion test is performed by each R-neighbor site I. Site I invites each of its neighbor j not in R to join R, if the number of reads that I received from j during the last period is greater the number of writes that I received during the same period from I itself or from a neighbor other than j. The contraction test is executed by each R-fringe site I. Site I requests permission from its neighbor site j in R to exit R, if the number of writes that I received from j during the last time period is greater than the number of reads that I received during this period. If site i is both an R-neighbor and an R-fringe, it executes the expansion test rest, and if the test fails (i.e., no site joins R), then it executes the contraction test. Finally, the switch test is executed, when R is a singleton test and the expansion test that the single site I in R has executed fails. Site I asks a neighbor site n to be the new singleton site, if the number of requests received by I from n during the last time period is larger than the number of all other requests received by i during the same Period. The ADR algorithm is shown to be

convergent-optimal in the following sense. Starting at any replication scheme, the algorithm converges to the replication scheme that is optimal to the current read-write pattern. The convergence occurs within a number of time periods that is bounded by the diameter of the network.

3.7 Forwarding Pointers

When the number of moves that a user makes is large relative to the number of calls it receives, it may be too expensive, and to update all database entries holding the user's location, each time the user moves. Instead, entries may be selectively updated and calls directed to the current location of a user through the deployment of forwarding pointers.

3.7.1 Two-Tier Architectures

In two-tier architectures, if the mobility of a mobile unit is high while it is located far way from its HLR, an excessive amount of messages is transmitted between the serving VLR and the HLR. Thus, to reduce the communication overhead as well as the query load at the HLR, the entry in x's HLR is not updated, each time the mobile unit x moves to a new location. Instead, at the VLR at x's previous location, a forwarding pointer is set up to point to the VLR in the new location. Now, calls to a given user will rest query the user's HLR to determine the rest VLR at which the user was registered, and then follow a chain of forwarding pointers to the user's current VLR. To bind the time taken by the lookup procedure, the length of the chain of forwarding pointers is allowed to grow up to a maximum value of K. An implicit pointer compression also takes place, when loops are formed as users revisit the same areas. Since the approach is applied on a per-user basis, the increase in the cost of call operations acts only the specie user. The router optimization extensions to IETF Mobile IP protocol include pointer forwarding in conjunction with lazy caching. The pointer forwarding strategy as opposed to replication is useful for those users who receive calls infrequently relative to the rate at which they relocate. Clearly, the beets of forwarding depend also upon the cost of setting up and traversing pointers relative to the costs of updating the HLR. An analytical estimation of the beets of forwarding is given in [31]. It is shown that under certain assumptions and if pointer chains are kept short (K <5), forwarding can reduce the total network cost by

20%-60% for users with call to mobility ratio below 0.5. A method for dynamically determining whether to update the HLR or not is proposed in the local anchoring scheme where a pointer chain of length at most one is maintained. For each mobile unit, a VLR close to it is selected as its local anchor (LA). In some cases, the LA may be the same as its serving VLR. Otherwise, the LA maintains a forwarding pointer to the current VLR of the mobile unit. For each mobile unit, the HLR maintains it's serving LA. To locate a mobile unit, the HLR is queried rest and then the associated LA is contacted. If the LA happens to be the serving VLR, no further querying is necessary, else the forwarding pointer is used to locate the mobile unit. Since after a call delivery the HLR knows the current location of a mobile unit, the HLR is always updated after a call to record the current VLR. Depending on whether the HLR is updated upon a move. Two schemes are proposed: static and dynamic local anchoring. In static local anchoring, the HLR is never updated at a move. In dynamic local anchoring, the serving VLR becomes the new LA if this will result in lower expected costs.

3.7.2 Hierarchical Architectures

To reduce the update cost, forwarding pointer strategies may be also deployed in the case of hierarchical architectures. In a hierarchical location scheme, when a mobile user x moves from zone I to zone j, entries for x are created in all databases on the path from j to LCA (j; I), while the entries for x on the path from LCA (j; I) to I are deleted. Using forwarding pointers, instead of updating all databases on the path from j through LCA (j; I) to I, only the databases up to a level m are updated. In addition, a forwarding pointer is set from node s to node t, where s is the ancestor of I at level m, and t is the ancestor of j at level m (Figure3.7). As in caching, the level of s and t can vary. In simple forwarding, s and t are leaf nodes, while in level forwarding, s and t can be nodes at any level. A subsequent caller reaches x through a combination of database lookups and forwarding pointer traversals. Take, for example, user x located at node 14 that moves to node 17 (Figure3.7). Let level m = 2. A new entry for x is created in the databases at nodes 17, 6 and 2, the entries for x in the databases at nodes 14 and 5 are deleted, and a pointer is set at x's entry in the database at node 1 pointing to the entry of x in the database at node 2. The entry for x at node 0 is not updated. When a user, say at zone 23, calls x, the search

message traverses the tree from node 23 up to the root node 0 where the rest entry for x is found, then goes down to 1, follows the forwarding pointer to 2, and traverses downwards the path from 2 to 17. On the other hand, a call placed by a user at 15, results in a shorter route: it goes up to 1, then to 2, and follows the path downwards to 17. Forwarding techniques can also be deployed for hierarchical architectures in which the entries of the internal nodes are actual addresses, rather than pointers to the corresponding entries in lower level databases. The example above is repeated in Figure3.8 for this case. Entries for x are updated up to level m = 2 and a forwarding pointer at leaf node 14 is set to redirect calls to the new location 17. Such architecture with internal nodes storing actual addresses rather than tree pointers is considered in where a performance analysis of forwarding is presented. Besides for warding, the scheme in also supports caching: leaf caching (i.e., caching the address of the called only at the zone of the caller) that is called jump updates and level caching (i.e., caching the address of the called on all nodes on the search path) that is called path compression. All combinations of forwarding (no forwarding (NF), simple forwarding (SF) and level forwarding (LF)) and of caching.

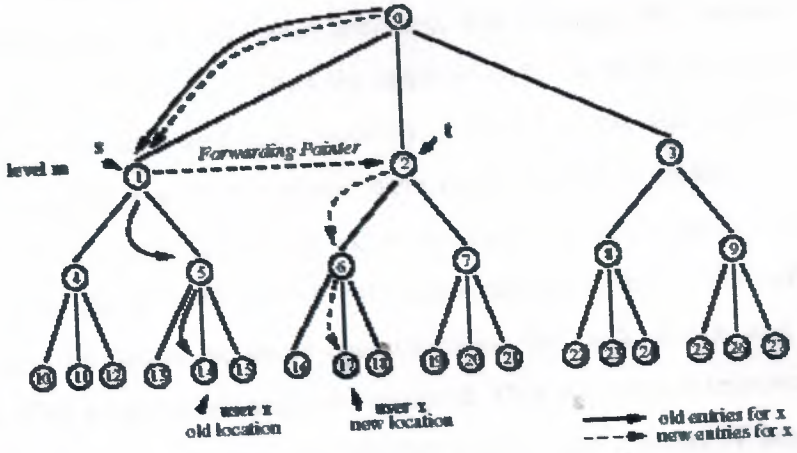

Figure3.7: Forwarding Pointers Example (entries are pointers to lower level databases)

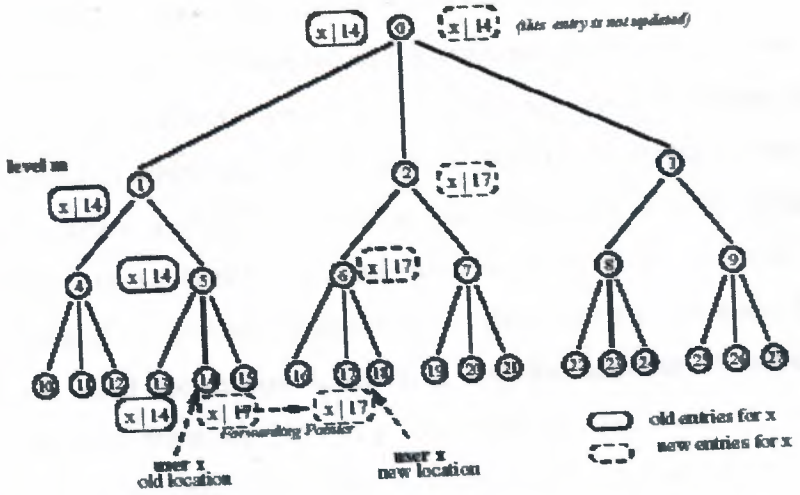

Figure 3.8: Forwarding Pointers Example (entries are exact addresses)

For the type (a) environment, the simulation showed the combination SF {PC to outperform all other combinations. The strategies using either NF or LF incurred a high cost of updates at each move. The SF {NU combination soured due to the very high search costs. Finally, the SF {JU did not perform well as the cached entries were not used very frequently since the calls were arbitrary. For the type (b) environment, SF {PC performed better as well except from the cases of high communication and low mobility and low communication and high mobility. In these cases, the combination SF {JU performed better because jump updates were more elective in reducing the search cost, since there was a specie set of callers. A per-user adaptive scheme was suggested to choose between the SF {PC and SF {JU combinations based on the call and mobility characteristics. To determine those characteristics, for each mobile unit a sequence is maintained of all moves made and calls received. This sequence determines the degree of mobility of the host (low or high) and whether it has a large number of frequent callers. Obsolete entries in databases at levels higher than m (e.g., the entry at node 0 in Figures 3.7 and 3.8) may be updated after a successful lookup. Another possibility for updates is for each node to send a location update message to the location servers on its path to the root during 0 peak hours. To avoid the creation of long chains of forwarding pointers, some form of pointer reduction is necessary. To reduce the number of forwarding pointers, a variation of caching is proposed in.

After a call to user x, the actual location of the user is cached at the rest node of the chain. Thus, any subsequent calls to x directed to the rest node of the chain use this cache entry to directly access the current location of x, bypassing the forwarding pointer chain. Besides, this form of caching that reduces the number of forwarding pointers that need to be traversed to locate a user, the database hierarchy must also be updated to avoid excessive look-up costs. Besides deleting forwarding pointers, this also involves the deletion of all entries in internal databases on the path from the rest node, I, of the chain *to the LCA of I and the current location*, j, and the addition of entries in internal databases on the path from the LCA to j. Take for example, chain *11! 18! 26! 14 that* resulted from user x moving from node 11, to nodes 18, 26, and 14, in that order. The entries for x at nodes 11, 18, and 26 are deleted. Then, the entries in higher-level databases leading to 11 are also deleted. In particular, the entry for x at 4 is deleted and entries are set at nodes 1, 5, and 14 leading to 14, the new location (see Figure3.9). Two conditions for initiating updates are proposed and evaluated based on setting a threshold either on the number of forwarding pointers or on the maximum distance between the rest node of the chain and the current location.

Forwarding pointer techniques nod applications in mobile software systems, to maintain references to mobile objects, such as in the Emerald System and in SSP chains.

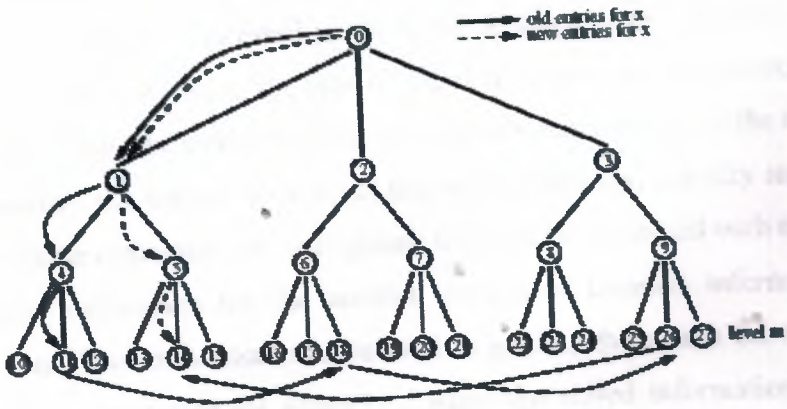

Figure3.9: Example of Pointer Purging

Object-based system in which objects can move within the system. SSP chains are chains of forwarding pointers for transparently migrating object references between processes in distributed computing. The SSP-chain short-cutting technique is similar to the simple update at calls method.

3.8 Taxonomy of Location Management Techniques

The techniques proposed in the previous sections are based on exploiting knowledge about the calling and moving behavior of mobile objects. Basically, two characteristics are considered: stability of calls and moves and locality of moves and calls. Stability in the case of calls means that most calls for each user originate from the same set of locations, for example, each user may receive most calls from a specie set of friends, family and business associates. Stability of moves refers to the fact that users tend to move inside a specie set of regions. For instance, they may follow a daily routine, e.g., drive from their home to their once, visit a predetermined number of customers, return to their once, and then back to their home. This pattern can change but remains axed for short periods of time. Locality refers to the fact that local operations are common. In particular, in the case of calls, a user frequently receives calls from nearby places, while in the case of moves, the user moves to neighbor locations more often than to remote ones. Another determinant factor in designing location techniques is the relative frequency of calls and moves expressed in the form of some call to mobility ratio. In general, techniques tend to decrease the cost of either the move or call operation in the expense of the other. Thus, the call to mobility ratio determines the easy of the technique. Figure3.10 summarizes the various techniques that exploit locality, stability and the call to mobility ratio. These techniques are orthogonal; they can be combined with each other. Besides developing techniques for the ancient storage of location information, the advancement of models of movement can be used in guiding the search for the current location of a mobile object (see for example,), when the stored information about its location is not current or precise. For instance, potential locations may be searched in descending order of the probability of the user being there.

An important parameter of any calling and movement model is time. The models should capture.

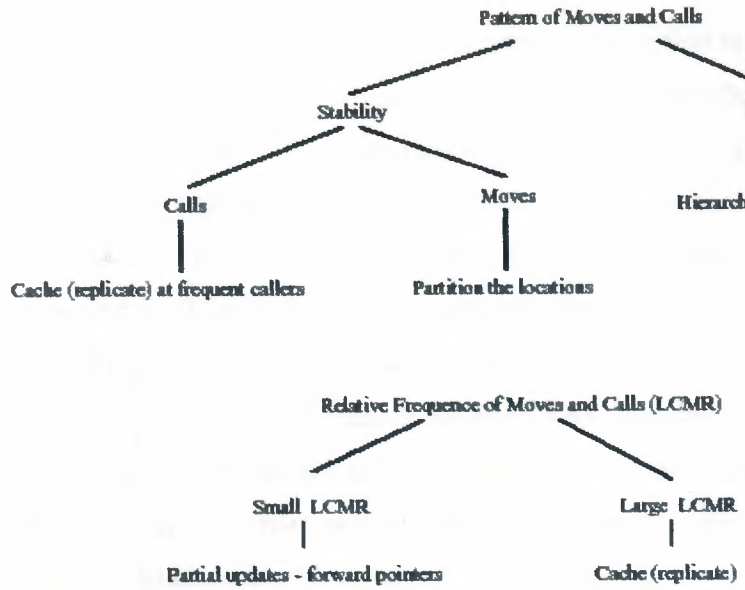

Figure3.10: Techniques along the Dimensions of Locality, Stability, and CMR (call to mobility ratio).

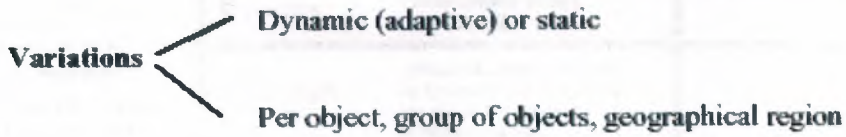

Figure3.11: Further Taxonomy of Location Techniques.

Temporal changes in the movement and calling patterns and their relative frequency as they appear during the day, the week or even the year. For instance, the track volume in weekends is die rent than that during a workday. Thus, dynamic adaptation to the current pattern and ratio is a desirable characteristic of location techniques. Another issue is the basis on which each location technique is employed. For instance, a specie location technique may be employed on a per user basis. Alternatively, the technique may be adopted for all PCS users or for a group of users based either on their geographical location (i.e., all users in a specie region), on their mobility and calling characteristics

(i.e., all users that receive a large number of calls) or a combination of both. Figure 3.11 summarizes these two dimensions of location techniques. Table 3.1 and 3.2 summarizes correspondingly the variations of the two-tier and hierarchical location scheme and their properties. Since the performance of most location techniques depends on the call to mobility ratio (CMR), in order for the system to adapt to the most appropriate technique based on the current CMR, dynamically estimating the current value of the CMR is a central issue. One approach to estimating CMRs is to calculate running estimates of CMRs on a per user basis. Two such strategies are proposed in. The running average algorithm maintains for every user the running counts of the number of incoming calls and the number of times that the user changes location. One problem with the running average algorithm is that estimations are taken from the entire past history of the user's movement and thus the algorithm may not be saliently dynamic to adequately react the recent history of the user's behavior.

Method	Variations		Applicable when:
Caching When x is called by y, cache x's location at y's zone	*Eager caching:*	Cache update overhead occurs at moves	Large LCMR Call Stability
	Lazy caching:	Cache update overhead occurs at calls	
Replication Selectively replicate x's address at the zones from which it receives the most calls	*Per-user Profile Replication:*	Additional constraints are set on the number of replicas per site and on the number of replicas per user	Large LCMR Call Stability
	Working Set:	Adaptive and distributed: the replication sites are computed dynamically by each mobile host locally	
Forwarding Pointers When x moves, add a forwarding pointer from its old to its new address	Restrict the length of the chain of forwarding pointers		Small LCMR

Table 3.2 Summary of Enhancements to the Basic Two-Tier Scheme. LCMR stands for the Local Call to Mobility Ratio.

The user movement process changes, a variation of this procedure, called the reset-K algorithm, and give more accurate estimations. With reset-K, running averages are estimated every K incoming calls. Another approach is to maintain information about the

CMR, for instance in the HLR, and download it during peak hours. Analytical estimations of the CMR are also possible. Finally, traces of actual moving users can be used (for example, the Stanford University Mobile Activity Traces (SUMATRA). Finally, another parameter that acts the deployment of a location strategy is the topology of network sites, how they are populated and their geographical connectivity. How the strategy scales with the number of mobile objects, location operation and geographical distribution is also an important consideration. Location strategies are evaluated based on two criteria, namely, the associated database and network overhead. In terms of database operations, various objectives are set including minimizing (a) the total number of database updates and queries, (b) the database load and size, and (c) the latency of each database operation. In terms of communication, location schemes aim at reducing among others (a) the total number of messages, (b) the number of hops, (c) the distance traveled, (d) the number of bytes generated, and (e) the sum of the track on each link or over all links.

3.9 Precision and Currency of Location Information

The focus of the previous sections was on evidently storing, updating and retrieving information about the location of moving objects. However, in some cases, to reduce the update cost, the stored information may not be precise in that it may cover more than one zone (cell). Then, to actually locate the user, after retrieving its stored location, a search is necessary inside all zones covered by

The stored location. Another possibility is that the stored location is not kept current, that is, it is not updated after each move.

Method	Issues/Variations	Appropriate when:
Caching When x at zone i is called by user y at zone j, cache at a node on the path from j to LCA(i, j) a pointer to a node on the path from i to LCA(i, j) to be used by any subsequent call to x from zone j.	Up to which tree level to maintain cache entries When to update cache entries	Large CMR Call Stability
Replication Selectively replicate x's location at internal and/or leaf databases.		Large CMR Call Stability
Forwarding Pointers When x moves from cell i to cell j, instead of updating all databases on the path from i to LCA(i, j) and from LCA(i, j) to j, update all databases up to some level m and add a forwarding pointer at the level m ancestor of i to point to the level m ancestor of j.	When and how to purge the forwarding pointers Setting the level m	Small LCMR
Partitions Divide the locations into sets (partitions) so that the user moves inside a partition frequently and crosses the boundary of a partition rarely. Keep information about the partition in which the user resides instead of its exact location.		Move Stability

Table3.3: Summary of Proposed Enhancements to Hierarchical Location Schemes.

3.9.1 Granularity of Location Information

The granularity of location information dyers with respect to how many location zones it covers. In the cellular architecture, this translates to how many and which cells are covered by each registration area. Then to locate a user all cells in the area are polled; a process called paging. There is a trade 0 in dining the granularity of a registration area. If it covers a small number of cells, the cost of updates is large, while if it covers a large number of cells, then the cost of searching increases. Dining the shape and size of each registration area is formulated as a combinatorial optimization problem. The objective is to minimize the location update cost subject to a constraint on the search cost to locate the user inside the registration area. Since, it turns out that the rectangular shapes are a good approximation to the optimum registration area shapes; the optimization problem is also stated for rectangular registration areas. The optimal registration area is calculated for each particular mobile unit or for each particular class of mobile units based on their respective mobility and call arrival patterns. The optimal registration area size is calculated for a mesh cell conjuration with square shaped cells given the costs of location updates and of searching inside a registration area. Each registration area consists of k

cells arranged in a square and the value of k is selected on a per-user basis. The work in uses a die rent model of mobility from the work in.

3.9.2 Frequency of Updates

So far, we have assumed that the stored information about the location of a moving object is updated each and every time the user moves. However, to reduce the update cost, the stored location information may be updated less frequently. Three strategies for initiating location updates are proposed: the time-based strategy, the movement-based strategy and the distance- based strategy. In the time-based update strategy, the stored location for each mobile user is updated periodically every T units of time. In the movement-based update strategy, the stored location is updated after the user has performed a preened number of movements across zones boundaries. Finally, in the distance-based update strategy, the stored location is updated when the distance of the stored location from the actual location of the user exceeds a preened value D. Analytical performance results show that the distance-based update approach outperforms the other approaches in most cases. However, distance based approaches are more decal to implement since they require knowing and computing a distance function. A die rent approach to signaling location updates is presented in. A subset of all cells is selected and designated as reporting cells. The location of a mobile user is updated only when it enters a reporting cell. The search to locate a mobile user is restricted to all cells that are in the vicinity of the reporting center to which the user last reported. For an arbitrary cellular topology, ending an optimal set of reporting cells is shown to be an NP-complete problem. Thus, optimal and near optimal solutions are advanced for special cases such as for the common topology of hexagonal cells. The reporting cells strategy is static in the sense that the set of reporting cells is xed. It is also global since the set of reporting cells is the same for all mobile users. A timer-based approach to location updates is developed in. A timeout parameter Tm is denned as the maximum amount of time to wait before updating the stored location given that the last stored location was m. The set of the time-out parameters Tm can be calculated by the system and communicated to the mobile users as necessary or calculated by the user directly. A distance-based update strategy is taken by the DOMINO (Databases for Moving Objects) project. In particular, a set of distance-

based update strategies, called dead-reckoning policies, are proposed that update the database location whenever the distance between the current location and the stored location exceeds a given threshold h. A cost model is developed to estimate the threshold h. The model takes into account the deviation and uncertainty in the estimation of the moving object's position as well as the communication cost of a location update. The deviation of a moving object at a particular time is the distance between the actual location of object x and the location of x stored in the database, e.g., one mile. The uncertainty of a moving object x is the size of the area in which the object x can possibly be, e.g., a circle with radius one mile. Both uncertainty and deviation have a cost or penalty in terms of incorrect decision making which is proportional to the size of the uncertainty and deviation respectively. In the speed dead-reckoning policy, the threshold is axed for each mobile object. In the adaptive dead reckoning policy, the threshold h is computed anew after each update so that it minimizes the cost until the next update. The disconnection detecting dead-reckoning policy considers the case in which for some reason the object is unable to generate updates. To avoid explicitly contacting the object, the threshold h is continuously decreasing as the tine interval from the last updates increases.

3.9.3 Search Procedures

When the registration area covers a number of possible locations or the stored location is not current, besides retrieving the stored location of the user, additional searching is necessary. The search procedure rest ideates the set of potential locations and then queries the locations in the set. The set of potential locations depends on the update policy and the granularity of the stored information. For instance, in the case of a distance-based strategy, all possible locations are in distance smaller that D from the stored location. Depending on whether we set any constraints on the delay or on the maximum number of locations that are polled before locating the mobile user, a search is called constrained or union- strained. The straightforward approach, also known as the \blanket polling" strategy is to query all potential location simultaneously. For the unconstrained case, it is shown in that given a probability distribution on user location; the search strategy that minimizes the expected number of locations polled is to query

each location sequentially in order of decreasing probability. It is also shown that this strategy substantially reduces the mean number of polling requests over the blanket approach even after moderate constraints are imposed. The results are extended in for the case where mobile units are allowed to move during the search procedure. It is shown that the optimal strategy is to search the conditionally most likely locations after each polling failure.

In a distance-based update strategy is adopted. An iterative algorithm is proposed based on dynamic programming for generating the optimal threshold distance D. Locations are searched in a shortest-distance-rest order such that locations closest to the location where the last location update occurred are queried rest. This an unconstrained searches; the delay to locate a mobile user is proportional to the distance traveled since the last location update. In constrained searching is considered for a distance-based update strategy. The delay to locate a user is constrained to be smaller than or equal to a preened maximum value. When a call arrives, the residing area of a mobile user is partitioned into a number of sub areas. These sub areas are then searched sequentially. The search in each sub areas is by blanket polling that is all locations in the sub area are simultaneously polled. By limiting the number of sub areas to a given value m, the time to locate a mobile user is smaller than or equal to the time required for the m polling operations.

3.10 Consistency and Recovery

The focus of this section is on consistency and recovery issues for location databases. Moves and calls are issued asynchronously and concurrently. Since each of them results in number of database operations, concurrency control is required to ensure correctness of the execution of these operations. In the case of a location database failure, database recovery is also required. We discuss recovery in the context of two-tier location schemes. Approaches to handling recovery in hierarchical schemes and their enhancements is an interesting, but less studied, research problem.

3.10.1 Concurrency Control

Since call and move operations arrive concurrently and asynchronously, concurrency control issues arise. If no special treatment is provided for concurrency, a call may read obsolete location data and fail to track the called. In this case, the call is lost and is reissued anew. This simple method does not provide any upper bound on the number of tries a call has to make before locating a moving user. Concurrency issues get more involved in hierarchical location schemes. In such schemes, a lookup operation results in a sequence of query operations issued at location databases at various levels in the hierarchy. Similarly, a move operation causes a sequence of update operations to be executed on various location databases. The underlying assumption so far was that moves and calls arrive sequentially and they are handled one at a time. Thus, it was assumed that there is no interleaving between the queries and the updates of the various call and move operations. This is a reasonable assumption only if all network and database operations are performed in negligible time. There are various approaches to the problem. For instance, setting at the old address a forwarding pointer to the new location is necessary to ensure that calls that were issued prior to the movement and thus arrive at the old address will not be lost. If a transactional approach is adopted, traditional database concurrency control techniques are used to enforce that each call and move operation is executed as a transaction, i.e., an isolated unit. This approach is highly impractical, since, for instance, acquiring locks at all distributed databases involved in a call or move operation causes prohibitive delays. A more practical approach is based on imposing a specie order on the way updates are per- formed. In particular, upon a move operation from I to j, rest entries at the path from j to LCA (I; j) are added in a bottom-up fashion and then the entries at the path from the LCA (I; j) to I are deleted in a top-down fashion. Special care must be given so that during the delete phase of a move operation, an entry at a level k 1 database is deleted only after servicing all lookups for higher-level databases. For an application of this approach to the regional matching method refer to and for an application to tree-structure architectures to. When a replication scheme is used, there is a need for deploying coherency control protocols, to maintain consistent replicas every time the user moves. Coherency control is a well-studied problem in transaction management. However, traditional approaches based on distributed locks or

timestamps may be expensive, thus other techniques that ensure a less strict form of replica consistency may be advanced. For example, if there is an HLR or a master copy that is always consistent, i.e., maintains the most up-to-date location, then a lookup can rely on this copy to locate the user when the location at a replica proves to be obsolete. Another approach is to use forwarding pointers at the old location to handle any incoming calls directed there from obsolete replicas.

3.10.2 Failure Recovery

Database recovery is required after the failure of a location database. In the case of the VLR/HLR the VLR, the HLR, or both may be periodically check pointed. If this is the case, after the failure the backup is restored. However, some of the records of the backup may be obsolete.

3.10.3 VLR Failure Restoration

If the VLR is check pointed, the backup record is recovered and used upon a failure. If the backup is obsolete, then all areas within the VLR must be paged to identify the mobile users currently in the VLR's zone. Thus, the restoration procedure is not improved by the check pointing process. The optimal VLR check pointing interval is derived to balance the checkpointing cost against the paging cost. GSM exercises periodic location updating: the mobile users periodically establish contact with the network to confirm their location. It is shown that periodic conurbation does not improve the restoration process, if the conurbation frequency is lower than 0.1 times of the portable moving rate. A mechanism is proposed, called location update on demand, which eliminates the need for periodic conurbation messages. After a failure, a VLR restoration message is broadcasted to all mobile users in the area associated with the VLR. The mobile users then send a conurbation message. To avoid congesting the base station, each such message is sent within a random period from the receipt of the request.

3.10.4 HLR Failure Restoration

In GSM, the HLR database is periodically check pointed. After a HLR failure, the database is restored by reloading the backup. If a backup record is obsolete, then when a call delivery arrives, the call is lost. The obsolete data will be updated by either a call origination or a location con- ration from the corresponding mobile user. An estimation of the probability of lost calls can be found. After a HLR failure, the HLR initiates a recovery procedure by sending a \Unreliable Roamer Data Directive" to all its associated VLRs. The VLRs then remove all records of mobile users associated with that HLR. Later, when a base station detects the presence of a mobile portable within its coverage area and the portable is registered at the local VLR, the VLR sends a registration message to the HLR allowing it to reconstruct its internal structures in an incremental fashion. Before the location is reconstructed, call deliveries to the corresponding mobile user are lost. A method called aggressive restoration is proposed in following this method, the HLR restores its data by requesting all the VLRs referenced in its backup copy to provide exact location information of the mobile users. The probability PU that the HLR fails to request information from a VLR is estimated. An algorithm is also proposed to identify VLRs that are not mentioned in the backup copy. These VLRs are such that there are portables that move in the VLR between the last HLR checkpointing and the HLR failure and do not move out of the VLR before the failure.

3.11 Querying Location Databases

Besides the ancient support of location lookups and updates, a challenging issue is the management of more advanced location queries. Examples of such queries include ending the nearest service when the service or the user is mobile (which is a form of a nearest-neighbor query), or identifying the route with the best track condition (which requires applying an aggregation operator to estimate the number of moving users in each route). Another application is sending a message to all users within a specie geographical area for example to perform geographically targeted advertising. Location queries may be imposed by either static or mobile users. In the case in which a single centralized DBMS is used to store the location of all moving objects, most research proposals follow the approach of building additional capabilities for handling moving objects on top of

existing Dbases. There is not much work on providing advanced query capabilities in distributed architectures. However, there is some very recent work on querying network directories that may be applicable to location directories as well.

3.11.1 Issues

A number of issues render processing location queries die rent from query processing in traditional database systems in both the centralized and the distributed case: The data values representing the location of mobile users are continuously changing. Besides a spatial dimension, querying location data has also a temporal dimension, thus an important issue is how to express and answer patio-temporal queries, for instance queries of the following form: what is the location of moving object x at time t. There are interesting queries that refer to future time, for example: nod all objects that will enter a specie region in the next hour". The answer to such queries is only tentative, that is it should be considered correct according to what is currently known. Location queries may include transient data that is data whose value changes while the queries are being processed, e.g., a moving user asking for nearby hospitals. another possible type of location queries are continuous queries, e.g., a moving car asking for hotels locating within a radius of 5 miles and requesting the answer to the query to be continuously updated. Issues related to continuous queries include when and how often should they be re-evaluated and the possibility of a partial or incremental evaluation.

An issue that complicates further the processing of location queries is the introduction of uncertainty, since to control the volume of location updates, the stored information about the location of a mobile object may be imprecise or out-of-date. Furthermore, in a variety of location queries, knowing the exact location of some users may not be necessary. Interesting question are :{ how to model and quantify imprecision in query answering, and {besides retrieving the stored locations, what is the optimal way to search to acquire the exact locations. The protocols for placing, replicating, caching and updating location data must be re-designed to evidently handle advanced queries in addition to workloads based on look-up and move operations. Since the number of moving objects may be large, to answer queries evidently, we would like to avoid examining the location of all objects. Thus, we would like to build an index on the location attribute. The type of index

depends on the architecture of the location databases. A patio-temporal query language, called FTL, with temporal operators that refer to the future has been proposed in FTL augments SQL with temporal (e.g., until, late) and spatial (e.g., inside-region) operators.

3.11.2 Centralized Database Architecture

Querying moving object databases has been discussed in the context of patio-temporal databases (for a survey on patio-temporal databases see for example in particular for indexing. Patio-temporal databases deal with geometries changing over time; that is with spatial objects whose position as well their extent (i.e., the region they cover) changes with time; queries refer to both the past and the future histories of moving objects. Here we focus on continuously moving objects having a zero extent. We focus on an important type of spatial queries called range queries. An example of a range query is \retrieve the objects that are currently inside a given region P". How such queries are processed depends on how the objects are modeled, and how they are stored and indexed. Modeling. To model the location of moving objects, a new data model, called MOST was in- traduced in. The novelty of most is the concept of a dynamic attribute, i.e., an attribute whose value changes continuously as a function of time without being explicitly updated. Location is modeled as a dynamic attribute. The value of the dynamic attribute depends on time t. for- molly, a dynamic attribute A is represented by three sub attributes: A. value, A.updatetime and a. function. A. function is a function of a single variable t that has value 0 at time t = 0. At time A.updatetime the value of A is A.value and until the next update of A, the value of A at time A.updatetime+t0 is given by A.value + A.function (t0) that is it changes with time according to f. An explicit update of the dynamic attribute may update any of its sub-attributes, e.g., update the function sub-attribute. The above model has been extended for the case in which mobile objects move on pre-specie routes. This is the case for example of airplanes or vehicles moving on a highway. In this case, three sub-attributes: A. route, A. direction and A. speed are used instead of the function attribute. A. route is a line spatial object denoting the route the object is moving on, A. direction is a binary indicator having value 0 or 1 indicating towards which endpoint of the route the object is moving, and A. speed is a linear function indicating the speed of the moving object. The model is also extended to include

information about the potential uncertainty and deviation of the stored location. Representing and Indexing Moving Objects. The indexing problem can be best described by decomposing it into two sub-problems. The rest problem concerns the geometric representation of the location attributes in multidimensional space. The issues involved are how to dine the multidimensional space and how to map the attributes of a moving object into a region (e.g., point, line) in this space. The object's region is not updated continuously but only when the attributes are explicitly updated. The second problem concerns developing an indexing method appropriate for the proposed representation. Existing spatial methods can be used; however, it is still unclear which one is more appropriate for the location distribution of mobile objects and for the specie geometric representation. First, assume that objects move on a 1-dimensional line, that is the location y of each object is described as a linear function of time, y (t) = v (t _ t0) + y0, where v is the velocity of the object and y0 the location of the object at time t0. Value-Time Representation and Indexing. This method plots the function y representing the way location changes with time. Thus, the horizontal-axis represents time (t) and the vertical-axis represents the value of location (y). An object is mapped to a trajectory that plots the location as a function of time. In fact, the trajectory is not a line but a semi-line starting at point (t0; y0). One way to index the lines is to use a spatial access method, for example each line could be approximated by a minimum bounding rectangle which is then indexed using an R-tree or a R*-tree. However, this approach is problematic. First, the corresponding minimum bounding rectangle covers a large portion of the space, whereas the actual space occupied by the line is small, thus leading to extremely large and overlapping rectangles. Second, it cannot represent innate objects well. Another approach is to decompose the data space into disjoint cells and store with each cell the set of lines it intersects. A drawback of this approach is that each line has many copies. This approach is taken in that use a quad tree-based index. The innate time dimension is partitioned into equal-sized time slices and an index is created for each slice. Theoretically, the union of these indexes is the master index of the whole time-value space being indexed. In practice, however, since the storage space is limited, when the period _T of an index is over, the index is disposed and the next index is generated. Thus, the index is reconstructed every _T time units; _T is called the index reconstruction period. An index

reconstruction algorithm is also proposed that is optimal in CPU and disk access overheads. Intercept-Slope or Dual Space Representation and Indexing. Consider an object whose location as a function of time is y (t) = a + ut, is called the intercept and u is called the slope. Then the representation space is constructed by the horizontal-axis representing the intercept and the vertical-axis representing the slope. Thus, the object is mapped to the point (a; u) in this space. The query region is transformed into a polygon. A number of indexing techniques are proposed and analyzed in. The problem becomes more decal if we consider moving objects in the plane. An important case is when objects move in the plane but their movement is restricted on using a given set of routes on the niter terrain. This is called the 1-5 dimensional problems in They propose representing each preened route as a sequence of connected line segments and indexing the positions of these line segments using a standard spatial access method. The full 2-dimensional problem is harder. In this case, in the value-time representation, the trajectories of moving objects are lines in the space.

The dual space representation is not directly applicable. One way to get the dual is to project the lines on the (x; t) and (y; t) planes and then take the dual (intercept-slope) representation for the two lines on these planes. Thus, now a line can be represented by a 4-dimensional point. The case is considered in which the trajectory of moving objects in the plane is obtained by discretely sampling the movement of objects in time and then using linear interpolation between these samples. Each line of the trajectory is then approximated by a minimum bounding box. An extension of the R-tree is proposed that keeps line segments that belong to the same trajectory together, i.e., in the same or neighbor nodes. This work does not address queries that refer to the future.

Uncertainty in Query Processing. Since the stored location of a moving object may deviate from its actual current location, there is some uncertainty in answering a query. Depending on the bound on the uncertainty of the stored location, it should be possible to calculate a bound on the uncertainty of the answer. The DOMINO project doer's two approaches, a qualitative and a quantitative one. In the qualitative approach, two kinds of semantics, namely the may and must semantics are incorporated. Under the May semantics, the answer to a range query is the set of all objects that are possibly inside the query polygon P, i.e., the objects whose uncertainty interval intersects P. Under the must

semantics, the answer is the set of all objects that are dentally inside P, i.e., the objects whose uncertainty interval are entirely inside P. In the quantitative approach, the answer is a set of objects each of which is associated with a probability that the object is inside P. To support such semantics, indexing should be extended. How to extend the value-time riper- sensation for the 1-dimensional case to support the may-must semantics is considered. In this representation, two lines are plotted for each object; one represents the maximum distance from y0 and the other the minimum distance from y0. Thus, at time t the value of location is an interval, the uncertainty interval, instead of a point. In this case, instead of being represented by a line or trajectory, an object is represented by a plane (the one between the two lines).

3.11.3 Distributed Database Architectures

There is not much research in querying distributed location directories. Query processing depends on the type of the architecture, for example in the case of hierarchical architectures, location databases are physically structured based on location. For example, an internal node in the hierarch chi contains location information for all mobile users currently in the geographical area it covers. Thus, it can be viewed as a distributed spatial index. Location queries in distributed architectures were introduced. In this approach, the architecture is based on partitions which are sets of locations between which the user relocates very often. A mobile user moves only infrequently to locations that belong to deferent partitions. The stored location information about a moving object is not its actual location but just the partition to which its actual location belongs. Thus, only movements among partitions generate database updates. The system guarantees bounded ignorance, in that the actual and stored location of a user is always in the same partition. To determine the actual location of a user, searching all locations in the partition of its stored location is necessary. Thus, deriving an optimal execution plan for a query involves determining an optimal sequence in which to search inside the partitions involved in the query. A tree-representation of this problem is proposed.

3.11.4 Service Discovery Protocols

With the wide-spread use of networking and the increasing number of network devises, there is a need for a scalable means to locate services. The location directories we have considered so far associate the name of a mobile object (service) with its location. Many recent approaches consider the problem of ending an appropriate object (service) by specifying a number of desired characteristics for the service.

The Service Location Protocol (SLP) provides an exile and scalable framework for providing hosts with access to information about the existence, location and conjuration of networked services. Traditionally, to locate a service, users provide the name of a network host (which is an alias for its network address) that supports the service. SLP eliminates the need for a user to know the name of a network host. Rather, the user supplies the desired type of service along with a set of attributes which describe the service. Based on this description, the SLP resolves the network address of the service for the user. Client applications are modeled as user agents and services are advertised by service agents. The user agent issues a service request on behalf of the client application specifying the characteristics of the service. The user agent receives a service reply specifying the location of all services in the network with the requested characteristics. The user agent may directly contact the service agents or in larger networks a directory agent. The directory agent functions as a cache. Service agents register the services they advertise in the directory agents. These advertisements must be refreshed or they expire. Services are grouped together using scopes. A scope may indicate a location, administrative grouping, and proximity in a network topology or some other category.

Interesting problems related to service location protocols include: Modeling services whose location change, e.g., how is the location of a moving service specie, storing, caching and replicating directory entries when either the services are mobile and/or the requests originate from mobile clients, updating directory entries and refreshing directory caches when the services are mobile and thus their location is fast changing, ancient location-aware querying, e.g., ending services based on location attributes, when the client requested the service, or the service is mobile: should the directory be hierarchically structured based on location or should an appropriate spatial index be built on top of it; what is a an appropriate index in this case; interoperability: how to relate

information available at deferent layers in the network, e.g., information stored at an HLR, to actually locate a service using a directory service protocol. Most of the above questions remain open. A hierarchical architecture for service discovery directory is proposed which are based on the use of a hash-based index. Finally, there has been some very recent research in incorporating database techniques in manipulating network directories including developing a data model and a declarative language for network directories and semantic caching of directory entries. Extending this work for the case of directories that include the location of moving objects is an interesting problem.

3.12 Summary

Managing the location of moving objects is becoming increasingly important as mobility of users, devices and programs becomes widespread. This paper focuses on data management techniques for locating, i.e., identifying the current location, of mobile objects. The exigency of techniques for locating mobile objects is critical since the cost of communicating with a mobile object is augmented by the cost of ending its location. Location management techniques use information concerning the location of moving objects stored in location databases in combination with search procedures that exploit knowledge about the objects' previous moving behavior. Various enhancements of these Techniques include caching, replication, forwarding pointers and partitioning. The databases for storing the location of mobile objects are distributed in nature and must support very high update rates since the location of objects changes as they move. The support of advanced queries involving the location of moving objects is a promising research topic.

4. MOBILE-AWARE ADAPTATION

.1Overview

Chapters four tells us about mobile aware adaptation it is about the networking section of mobile system between users and severs. And by using the internet we communicate and we send the information as in file format. The files are transferred. We have client who did this prospect. And the HTTP protocol is stateless, requiring that each request contain the browser's capabilities. For a given browser, this information is the same for all requests.

4.2Introduction

Mobile clients could face wide variations and rapid changes in network conditions and local resource availability when accessing remote data. In order to enable applications and systems to continue to operate in such dynamic environments, the mobile client-server system must react by dynamically adjusting the functionality of computation between the mobile and stationary hosts. In other words, the computation of clients and servers has to be *adaptive* in response to the changes in mobile Environments the range of strategies for application and system adaptation is identified,

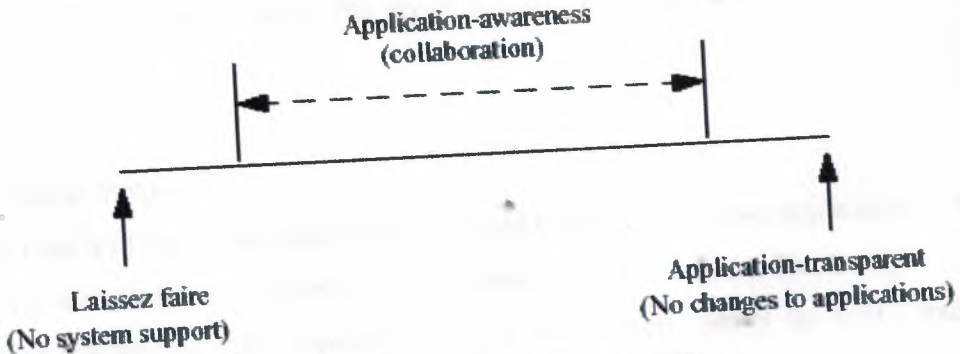

Figure 4.1 Range of adaptation strategies

The range is delimited by two extremes. At one extreme, adaptation is entirely the responsibility of individual Applications. This approach, called lassie-faire adaptation, avoids the need for system support. The other extreme, called application-transparent

ɔtation, places the entire responsibility for. Adaptation on the system. A typical case this approach is to use proxies to perform adaptation on behalf of applications. ween these two extremes lies a spectrum of possibilities that are referred to as lication-aware adaptation. This approach supports collaborative adaptation between applications and the system. That is, the applications can decide how to best adapt to changing environment while the system provides support through the monitoring of sources and the enforcing of resource allocation decisions. This section will discuss fferent proposed adaptation approaches.

3 Applications-Transparent Adaptation

Many existing client-server applications are built around the assumption that the nvironment of a client does not change. These applications are usually unaware of the nobility and make certain Assumptions about the resource availability. The approach of application- transparent adaptation attempts to make these applications work with no modification in mobile environments. This is done by having the system shield or hides the differences between the stationary and mobile environments from applications. Examples of this approach include Coda Little Work and Web Express. In these examples, a local proxy runs on the mobile host and provides an interface for regular server services to the applications. The proxy attempts to mitigate any adverse effects of mobile environments.

4.4 File System Proxy.

The basic idea is to use a file system proxy to hide mobile issues from applications and to emulate file server services on the mobile computers (see Figure 4.2). The Coda File system pioneering this approach, uses a file system proxy to make existing applications work with no modification.

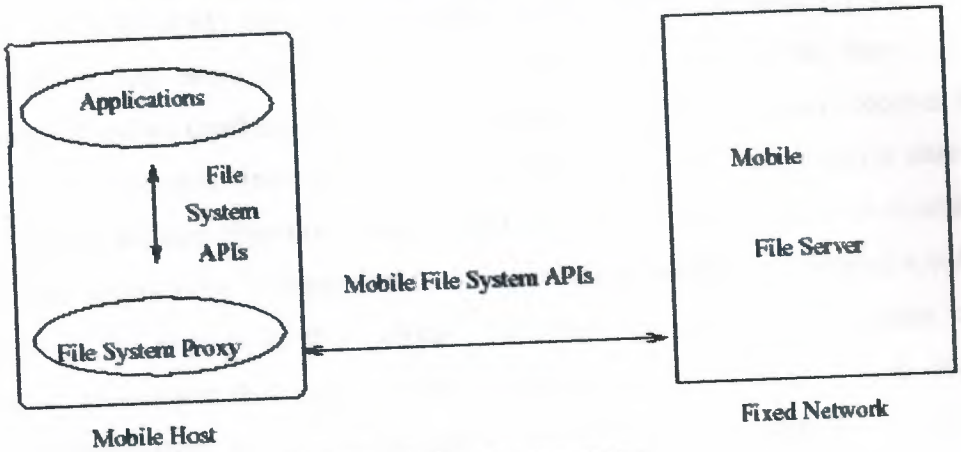

Figure 4.2 File system proxy

The proxy logs all updates to the file system during disconnection and replays the log on reconnection. Automatic mechanisms for conflict resolution using optimistic concurrency control are provided for directories and files through the proxy and the file server. The file system proxy in Coda facilitates the following features: Disconnected Operations: A small collection of trusted Coda servers exports a location-transparent UNIX file name space to a larger collection of entrusted clients. On each client, a user-level process, Venus, manages a file cache on the local disk. Venus acts as a file system proxy and bears the brunt of disconnected operations. Venus operates in one of three states: hoarding, emulating, and reintegrating. In the hoarding state, server files are pre-fetched onto the mobile computer. Upon disconnection, Venus enters the emulating state and begins logging updates in a client modify log. In this state, Venus performs log optimizations to improve performance and reduce resource usage. Upon reconnection, Venus enters the reintegrating state, where it synchronizes its cache with the servers, propagates updates from the client modify log, and returns to the hoarding state. In anticipation of disconnection, users may hoard data in the cache by providing a prioritized list of files in a per- client hoard database. Venus combines the hoard database information with LRU (Least Recently Updated) information to implement a cache management policy. Periodically, Venus walks the cache to ensure that the highest

priority items are present and consistent with the servers. A user may also explicitly request a hoard walk at any time. Since consistency is based on optimistic replica control, update conflicts may occur upon reintegration. The system ensures the detection and confinement of update conflicts and provides mechanisms to help the users recover from them. Weakly Connected Operations: The file system proxy pre-fetches server data into the client cache and uses object or volume1 callbacks for the cache validation in order to support weak connectivity. Volume call back is pessimistic in that invalidating a volume invalidates all the objects in this volume. However, the gain is in reducing cache invalidation information that needs to be communicated between the client and the server. The file system proxy can determine, based on factors such as cached data structures and connectivity changes, whether object or volume callbacks are best for a particular connection. The variable granularity of callback attempts to minimize the cost of Validation and invalidation to provide effective support of operations for weakly connected clients. Isolation-only Transactions: Disconnected operations may result in data inconsistency due to conflicting operations on multiple disconnected computers. Isolation-only Transaction (IOT) is proposed to automatically detect read/ write conflicts. The execution of IOT is realized by the file system proxy code in the Coda system. An IOT provides consistency guarantees depending on the system connectivity conditions. Unlike traditional transactions, it does not guarantee failure atomicity and only conditionally guarantees permanence. When an IOT is completed, it enters either the committed or the pending state, depending on the connectivity condition (see Figure 4.3). If the execution of an IOT does not contain any partitioned file access, it is committed and its result is made visible on the servers. Otherwise, it enters the pending state for later validation.

The result is temporarily held within the client's local cache and is visible only to subsequent processes on the same client. When the relevant partitions are repaired, the IOT is validated according to the isolation consistency criteria, namely, serializability. If the validation succeeds, the result will be immediately reintegrated and committed to the servers. Otherwise, the IOT enters the resolution state. When it is automatically or man-

1A volume is a collection of related objects. Dually resolved, it will commit the new

result to the server. In addition to the Coda project, other projects address similar adaptation issues for mobile file system applications. In the Rover project, a file system proxy is added to the Rover Toolkit's object based model. It allows the Rover Toolkit to support both a file model and an object model for mobile applications. The file system proxy in Rover Toolkits also addresses a number of issues related to file caching, perfecting, and conflict detection and resolution. These issues are similar to those addressed by the Coda file system, except that they are associated with integrating a file system model with an object-based model. The Rover file system proxy consists of two components: a user-level installable local file system proxy located on the client and a remote file system proxy running on a Rover server. The two components work together with the use of Rover's queued communication mechanism that supports automatic message compression and batching. The Focus file system is another file system supporting disconnected operations with application transparent adaptation, but relies on version vectors to detect conflicts. The Little Work project caches files for smooth disconnection from an AFS file system. Conflicts are detected and reported to the user for manual resolution. Web Proxy. Web proxy enables Web browsing applications to function over wireless links without imposing changes on browsers and servers. Web proxy can be used to prefect and cache Web pages to the mobile client's machine, to compress and transform image pages for transmission over low-bandwidth links, and to support disconnected and asynchronous

- ***J. Jing et al.***

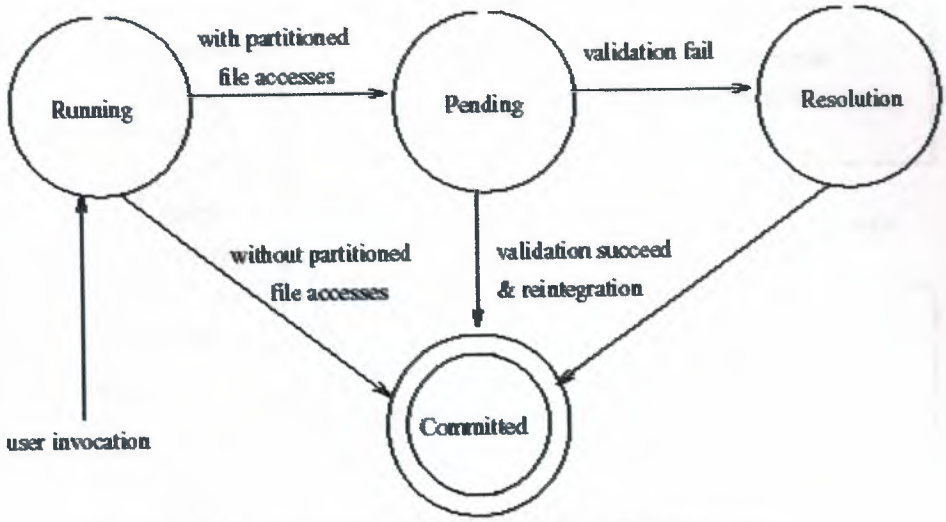

Figure 4.3 A state transition diagram for IOT execution

Browsing operations Web Express uses this approach to intercept and control communications over the wireless link for the purposes of reducing traffic volume and optimizing the communication protocol to reduce latency. Two components are inserted into the data path between the Web clients and the Web server: the Client Side Intercept (CSI) process that runs in the client mobile device and the Server Side Intercept (SSI) process that runs within the wired and fixed network (see Figure 4.4). The CSI intercepts HTTP requests and, together with the SSI, performs optimizations to reduce bandwidth consumption and transmission latency over the wireless link. From the viewpoint of the browser, the CSI appears as a local Web proxy that is co resident with the Web browser. On the mobile host, the CSI communicates with the Web browser over a local TCP connection via the HTTP protocol. Therefore, no external communication occurs over the TCP/IP connection between the browser and the CSI. No changes to the browser are required other than specifying the (local) IP address of the CSI as the browser's proxy address. The CSI communicates

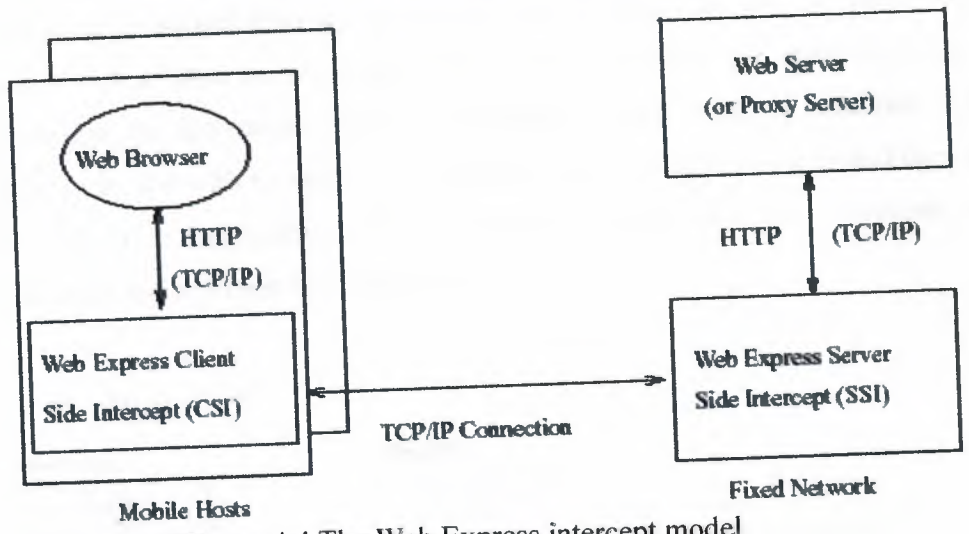

Figure 4.4 The Web Express intercept model

With an SSI process over a TCP connection using a reduced version of the HTTP protocol. The SSI reconstitutes the HTML data stream and forwards it to the designated CSI Web server (or proxy server). Likewise, for responses returned by a Web server (or a proxy server), the CSI reconstitutes an HTML data stream received from the SSI and sends it to the Web browser over the local TCP connection as though it came directly from the Web server. The proxy approach implemented in Web Express offers the transparency advantage to both Web browsers and Web servers (or proxy servers) and, therefore, can be employed with any Web browser. The CSI/SSI protocols facilitate highly effective data reduction and protocol optimization without limiting any of the Web browser functionality or interoperability. Web Express optimization methods are summarized below: Caching: Both the CSI and SSI cache graphics and HTML objects. If the URL specifies an object that has been stored in the CSV's cache, it is returned immediately as the response. The caching functions guarantee cache integrity within a client-specified time interval. The SSI cache is populated by responses from the requested Web servers. If a requested URL received from a CSI is cached in the SSI, it is returned as the response to the request. Differencing: CSI requests might result in responses that normally vary for multiple requests to the same URL (e.g., a stock quote server). The concept of differencing is to cache a common base object on both the CSI

and SSI. When a response is received, the SSI computes the difference between the base object and the response and then sends the difference to the CSI. The CSI then merges the difference with its base form to create the browser response. This same technique is used to determine the difference between HTML documents. Protocol reduction: Each CSI connects to its SSI with a single TCP/IP connection. All requests are routed over this connection to avoid the costly connection establishment overhead. Requests and responses are multiplexed over the connection.

- *J. Jing et al.*

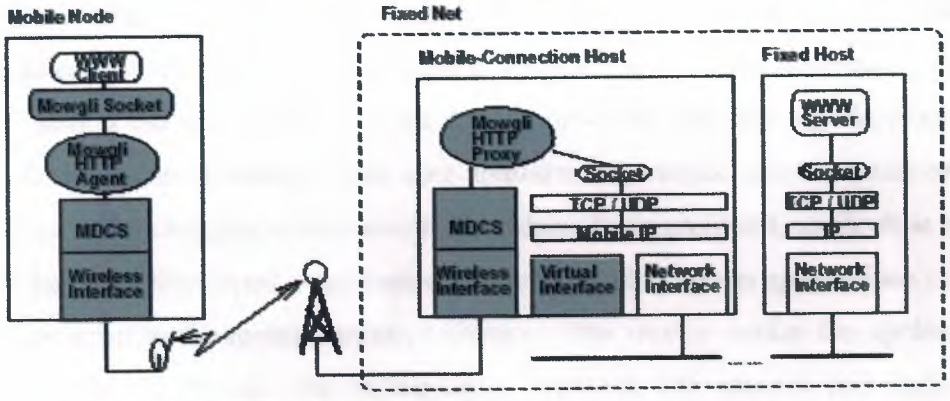

Figure 4.5 The Mowgli wireless Web browsing architecture

Header reduction The HTTP protocol is stateless, requiring that each request contain the browser's capabilities. For a given browser, this information is the same for all requests. When the CSI establishes a connection with its SSI, it sends its capabilities only on the first request. This information is maintained by the SSI for the duration of the connection. The SSI includes the capabilities as part of the HTTP request that it forwards to the target server (in the wire line network). The Mowgli project also uses a similar approach to support Web applications over wireless links (see Figure 4.5). A specialized HTT pageant and a specialized HTTP proxy are applied to: (1) reduce unnecessary message exchange, (2) reduce the volume of data transmitted over the wireless link, and (3) support disconnected operations. The HTTP agent acts as a local Web proxy on mobile client hosts while the HTTP proxy is located in the fixed network. With the agent-proxy

approach, neither Web clients nor servers need to be modified. The HTTP agent intercepts requests generated by the Web client on the mobile node. It communicates with the HTTP proxy in the fixed network. Both the agent and the proxy maintain a cache of their own to improve performance over low-bandwidth links.

4.5 Applications-Aware Adaptation

The approach of application-transparent adaptation does not require changing existing applications to run in mobile environments. However, it could sacrifice functionality and performance. As applications are shielded from dealing with mobility, it might be very hard for the system to make adaptation decisions that meet the needs of different and diverse applications. As a result, it may have to require some manual intervention by the user (e.g., having the user indicate which data to pre-fetch onto the mobile device) to make applications run smoothly. Such user-administered manual actions could be less agile to adapt to the changing environment. To address these problems, application aware adaptation has been developed. Application-aware adaptation allows applications or their extensions to react to the mobile resource changes. One way to realize the application-aware adaptation is through the collaboration between the system and individual applications. The system monitors resource levels, notifies applications of relevant changes, and enforces resource allocation decisions. Each application independently Decides how best to adapt when notified. In a video player application, for example, such adaptation allows the video player system to scale back quality (and resource consumption) when application performance is poor and to attempt to discover additional resources by optimistically scaling up usage, Depending on where the adaptive application logic resides, the approaches of application-aware adaptation can be divided into the following categories: client- based application adaptation, client-server application adaptation, and proxy-based application adaptation. The client-based adaptation allows the applications on mobile clients to react to the environmental changes, while client- server adaptation might have applications on both client and server to adapt to the changes. The proxy-based Adaptation supports application-specific adaptation on the proxy server in the fixed networks. The application-specific proxies

have been used as an intermediary between existing servers and heterogeneous mobile clients. The proxies can perform aggressive computation and storage on behalf of mobile clients. These approaches can be complementary for a client-server information system. For example, both client-based and proxy-based adaptation can be used together in a single system to deal with the mobility.

4.6 Client-Based Application Adaptation

The Odyssey project demonstrates a client-based collaborative adaptation approach for applications on mobile clients. In the collaborative adaptation, the system provides the mechanisms of adaptation, while the applications are free to specify adaptation policy. The division of responsibility addresses the issues of application diversity and concurrency. In a mobile information system, application data can be diverse in terms of data formats and consistency requirements. For example, the application data may be stored in one or more general-purpose repositories such as file servers, SQL servers, or Web servers. Alternatively, it may be stored in more specialized repositories such as video libraries, query-by-image-content databases, or back ends of geographical information systems. The application data can also have different dimensions for the specification and representation. For example, Video data can have at least two specification dimensions: frame rate and image quality of individual frames. Spatial data, such as topographical maps, has dimensions of minimum feature size and resolution. Furthermore, concurrent applications can be very useful for mobile users. For example, an information filtering application may run in the background monitoring data such as stock prices and alert the user as appropriate. The collaborative adaptation accommodates the application diversity by allowing applications to determine how application data presented at a client matches the reference copy at the server based on resource levels. It also supports the application concurrency by allowing the system to retain control of resource monitoring and arbitration. The application-aware adaptation in Odyssey is performed through the use of type-specific operations between the system and applications. The type awareness is incorporated into both the system for efficient resource usage and the applications for differential handling of data types. The system-

level knowledge of data types facilitates the optimization of the resource usage for different and diverse applications. For example, the size distribution and consistency requirements of data from an NFS server differ substantially from those of relational database records. Image data may be highly compressible using one algorithm, but not another. Video data can be efficiently shipped using a streaming protocol that drops rather than retransmits lost data; in contrast, only reliable transmissions are acceptable for file or database updates. Odyssey incorporates type-awareness via specialized code components called wardens. A warden encapsulates the system-level support at a client to effectively manage a data type. To fully support a new data type, an appropriate warden has to be written and incorporated into Odyssey at each client. The wardens are subordinate to a type-independent component called the viceroy, which is responsible for centralized resource management. The collaborative relationship in the application-aware adaptation is thus realized in two parts. The first, between the viceroy and its wardens, is data-centric: it defines the consistency levels for each data type and factors them into resource management. The second, between applications and Odyssey, is action-centric: it provides applications with control over the selection of consistency levels supported by the wardens.

A number of similar approaches have also been discussed in the literature. In the Prayer system the application-aware adaptation is supported with the use of abstractions: Quos classes and adaptation blocks. A Quos class is defined by specifying the upper and lower bounds for resources. An application divides its execution into adaptation blocks. An adaptation block consists of a set of alternative sequences of execution, each associated with a Quos class. At the beginning of an adaptation block, an application specifies the Quos classes that it is prepared to handle, along with a segment of code associated with each class and an action to take should the Quos class be violated within the code segment. In Welling and Baronet, application- aware adaptation is implemented under an event delivery framework. In the framework, a notification subsystem, called event channel, delivers different events that are generated by the environment monitor to applications based on delivery policies. For example, a low memory event may be delivered to each application in series until enough memory is freed for other uses, while

a low bandwidth event can be delivered to each application in parallel. The applications are notified of the events to react to the environmental changes.

4.7 Client-Server Application Adaptation

The Rover Toolkit supports the application aware adaptation through the use of releasable dynamic object (RDO). The key task of the programmer when implementing an application-specific adaptation with Rover is to define Rods for the data types manipulated by the application and for the data transported between the client and the server. The programmer divides the program that contains the Rods into portions that run on the client and portions that run on the server; these parts communicate by means of queuing RPC.

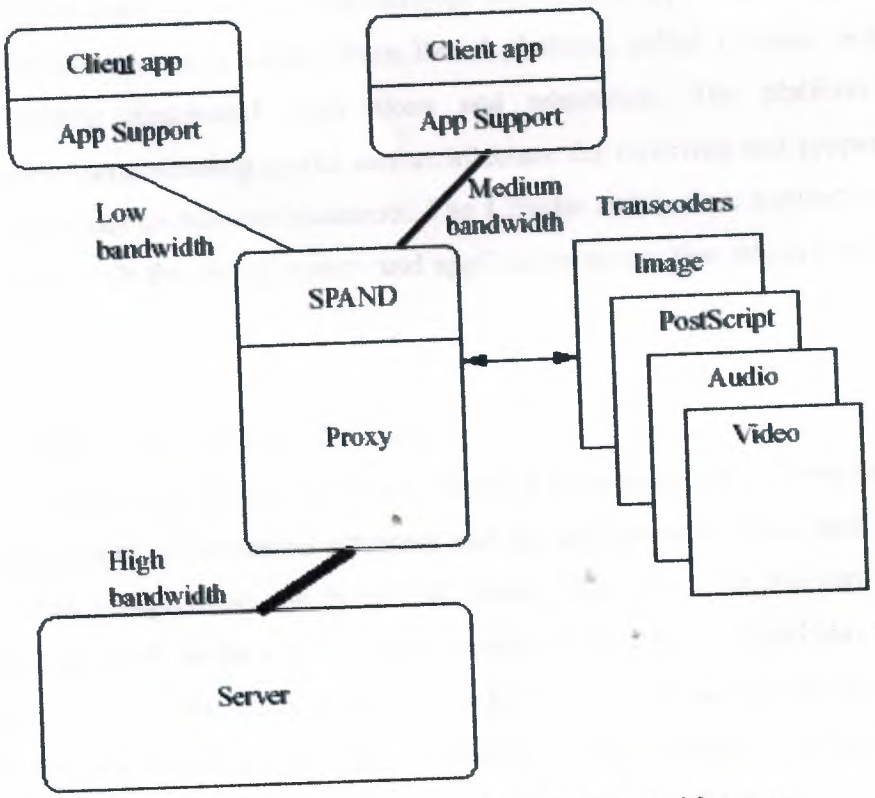

Figure 4.6 A proxy based adaptation architecture

The programmer also defines the methods that update objects, including code for conflict detection and resolution. At the level of RDO design, application designers have semantic knowledge that is very useful in implementing application- aware adaptation. By tightly coupling data with program code, applications can manage resource utilization more carefully than possible with a classic file or object system that handles only generic data. For example, an RDO can include compression and decompression methods along with compressed data in order to obtain application- specific and situation-specific compression, reducing both network and storage utilization. The RDO approach provides a generic framework to implement type-specific or application specification aerations for application aware adaptation. The application that consists of these RDO modules actively cooperates with the runtime system of the Rover Toolkit to import Rods onto the local machine, invoke well-defined methods on those objects, export logs of method invocations on Rods to servers, and reconcile the client's copies of the objects with the server's. In Davis et al. a topple space Based platform, called L2imbo, is proposed to support mobile distributed applications and adaptation. The platform offers an asynchronous programming model and architecture for reporting and propagating QOS information about mobile environments. The L2imbo architecture supports mechanisms of adaptation with the use of system and application agents that interact with the topple space.

4.8 Proxy-Based Application Adaptation

The application-specific proxy has been proposed as an intermediary between clients and servers to perform computation intensive and storage intensive tasks, such as data type specific loss compression, on behalf of client. This proxy architecture reduces the bandwidth demands on the infrastructure through loss compression and allows legacy and other nonstandard. The proxy-based application adaptation allows the proxy agents to react to the environmental changes on behalf of mobile clients. This approach avoids inserting adaptation machinery at each origin server. From the client's perspective, the proxy is simply a server that gets the data from someplace else. In the BARWAN project, the application specific proxy uses the transponders to optimize the quality of service for

the client in real time (see Figure 4.6). To use transcending to adapt to network variation, the proxy must have an estimate of the current network conditions along the path from the proxy to the client. SPAND (Shared Passive Network Performance Discovery), a network measurement System, allows a measurement host to collect the actual application-to-application network performance (e.g., available bandwidth and latency) between proxies and clients. SPAND monitors end-to-end bandwidth and connectivity to the clients (and servers) and notifies the proxy of any changes, which may result in changes in transcending to adjust the quality of service. The original servers are unaware of the transformations or of the limited capabilities of the clients or networks. Compact HTML is a W3C submission of a standard for small information appliances. It defines a subset of HTML for small information appliances, such as smart phones, smart communicators, and mobile Pads. The standard is intended as guidelines for manufacturers of small information devices, service providers, carriers, and software developers. Another similar language is the Hand-held Device Markup Language. The HDML standard has been adapted into the Wireless Markup Language (WML), which makes up the application and presentation layers of the Wireless Application Protocol (WAP) stack, whose specification is being developed by the WAP forum as a de facto standard. WAP uses WML and provides third party proxies (via a session-layer protocol) as a means to negotiate device capabilities in terms of content characteristics. The WAP architecture is shown in Figure 4.7

J. Jing et al.

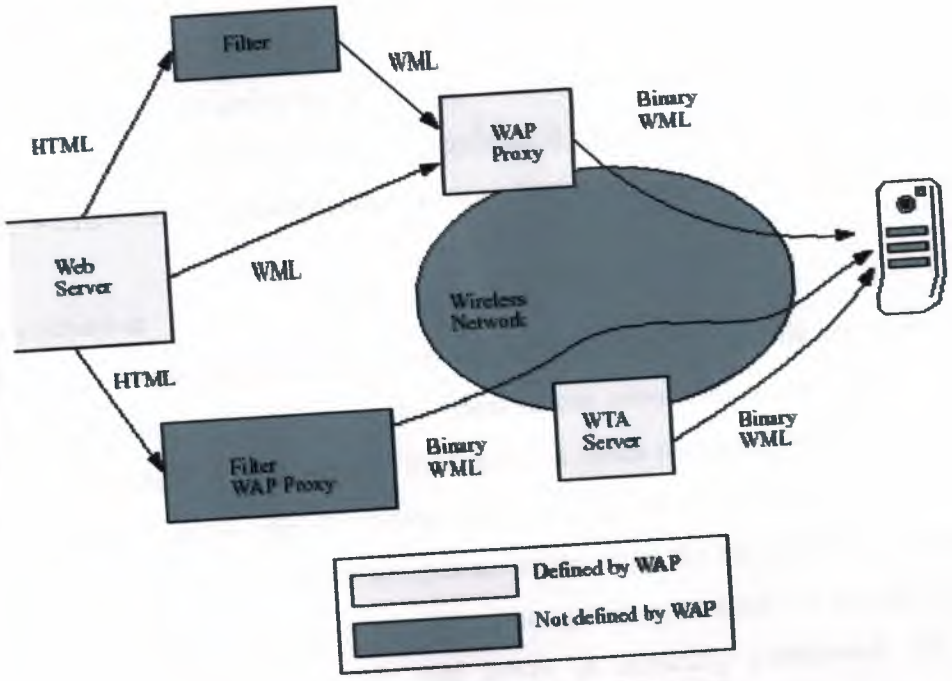

Figure 4.7 The WAP architecture

The WAP proxies are capable of adapting to the mobile devices by negotiating and filtering the HTML Web content. Filtering is performed as a translation to and from a compact subset of the HTML language. The filtering process itself can be prescribed or automated, based on capability negotiation.

4.9 Summary

The summary of chapter five tells us that the communication and networking section between the client and sever and the transferring the files or information through the internet and makes up the application and presentation layers of the Wireless Application Protocol. The application-specific proxy has been proposed as an intermediary between clients and servers to perform computation intensive and storage intensive tasks, such as data type specific loss compression, on behalf of client. The Odyssey project demonstrates a client-based collaborative adaptation approach for applications on mobile clients.

5. EXTENDED CLIENT-SERVER MODEL

5.1Overview

In chapter five we are going to explain about thin client architecture, full client flexible client collaborative architectures and application about specific proxy virtual mobile server. We have some models to show relation between host and the network.

5.2Introduction

Another way to characterize the client-server computing in mobile environments is to examine the effect of mobility on the client-server computing model. In a client-server information system, a server is any machine that holds a complete copy of one or more databases. A client is able to access data residing on any server with which it can communicate. Classic client-server systems assume that the location of client and server hosts does not change and the connection among them also does not change. As a result, the functionality between client and server is statically partitioned. In a mobile environment, however, the distinction between clients and servers may have to be temporarily blurred resulting in an extended client-server model shown in Figure5.1. The resource limitations of clients may require certain operations normally performed on clients to be performed on resource-rich servers. Conversely, the need to cope with uncertain connectivity requires clients to sometimes emulate.

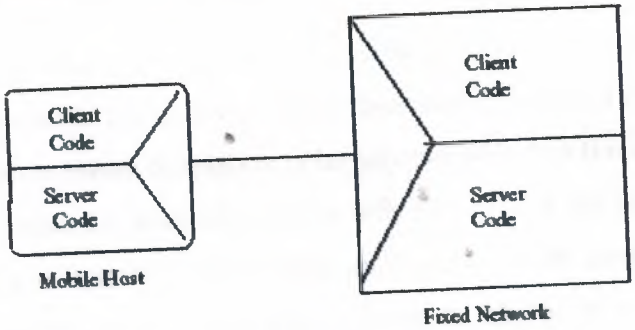

Figure 5.1 Extended client server model

ACM Computing Surveys, the functions of a server. An extreme case is called the thin client architecture that offloads most application logic and functionality from clients to stationary servers. In the thin client architecture, applications in stationary servers are

usually mobile-aware and optimized for mobile client devices. The thin client architecture is especially suitable for dumb terminal or small PDA applications. The other extreme case is the full client architecture. The full client architecture emulates server functions on the client devices and, therefore, is able to minimize the uncertainty of connectivity and communications.

5.3 Thin Client Architecture

The Info Pad project demonstrates the approach of thin client architecture. The Info Pad system is composed of four layers shown in Figure5.2: Pad, Info Net, Type Servers, and Applications. The Pad is a low power, portable multimedia terminal that is capable of displaying text and graphics, playing audio and compressed video, recording audio and capturing pen input. The Info Net layer presents the software layer above the Pad with an abstraction in which each pad appears as a stationary, network-connected, multimedia terminal. This layer contains the routing algorithms to make mobility seamless and the routines to manage the wireless network resources. The type servers make the pad appear as a typical workstation to applications; this allows for compatibility with standard workstation software. The type servers shield applications from knowledge of the mobile environments and terminal hardware. However, applications are optimized for use on the pad. The optimization takes advantage of the special characteristics of the pad, such as small screen size, lack of a keyboard, and support for handwriting and speech recognition.

While applications in the Info Pad system are customized for the characteristics of the pad, they do not contain code that allows them to dynamically adapt to changes in mobile networks. Instead, mobile-aware adaptation is largely performed in the Info Net and type server layers. In this sense, the adaptation in the Info Pad system can be characterized as application-transparent adaptation rather than application-aware adaptation. The thin client architecture from CITRIX Corporation allows a variety of remote computers, regardless of their platform, to connect to a Windows NT terminal server to remotely access a powerful desktop and its applications. A server called Metairie runs under Windows NT in the desktop machine and communicates with the thin clients executing at the remote computers using the Independent Computing Architecture protocol (ICA).

The ICA client and the Metairie server collaborate to display the virtual desktop on the remote computer screen. They also collaborate to process mouse and keyboard events, and to execute programs and view data stored at the server. All executions are remote and none takes place at the client portable computer. A research project at Motorola extended Cyrix's thin client architecture so that it is optimized in the wireless environment. The work pointed out that bandwidth limitation is not as detrimental to the thin client performance as network latency. This is because the thin clients' use of bandwidth is limited. W4 applies the technique of dividing application functionality between a small PDA and a power- Fixed

5.4 Full Client Architecture

Mobile clients must be able to use networks with rather unpleasant characteristics: intermittence, low bandwidth, high latency, or high expense. The connectivity with one or more of these properties is referred to as weak connectivity. In the extreme case, mobile clients will be forced to work under the disconnected mode. The ability to operate in disconnected mode can be useful even when connectivity is available. For example, disconnected operations can extend battery life by avoiding wireless transmission and reception. It can reduce network charges, an important feature when charge rates are high. It allows radio silence to be maintained, a vital capability in military applications. Finally, it is a viable fallback position when network characteristics degrade Beyond usability. Full client architecture can be used to effectively support the disconnected or weakly connected clients. Compared to thin client architecture, the full client architecture is at the other extreme of the range of extended client-server model. The full client architecture supports the emulation of functions of servers at the client host so that applications can be executed without fully connecting to remote servers. The emulation can be supported through a proxy or a "lightweight" server residing on client hosts. For example, a proxy can emulate some database operations to allow mobile users to work in a disconnected mode. Systems, enabling the full client architecture, include CODA
And Web Express.

- ***J. Jing et al.***

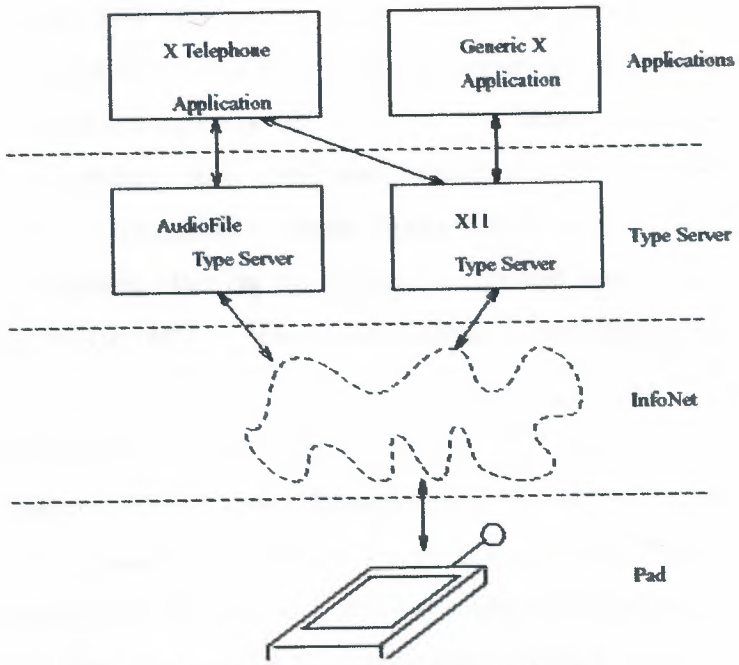

Figure 5.2 Info Pad system Layering

5.5 Flexible Client-Server Architecture

Flexible client-server architecture generalizes both thin client and full client architectures in that the roles of clients and servers and application logic can be dynamically relocated and performed on mobile and stationary hosts (see Figure 5.3). In the flexible architecture, the distinction between clients and servers may be temporarily blurred for purposes of performance and availability. Furthermore, the connection between clients and servers can be dynamically established during the execution of applications (e.g., for the service handoffs).Mobile Objects (Also known as mobile agents) are programming entities that can freely roam the network. Mobile objects enable client functions to run not only on mobile hosts, but also on stationary hosts as well. Furthermore, mobile objects allow clients to download the server code to the mobile host for execution. Mobile objects can contain threads and, therefore, be active. They can maintain state information and make intelligent decisions using it. They differ from downloadable applets in that they can independently roam among different machines and are not limited to being downloaded once from server to client. Agents (developed at Dartmouth

personal laptops, should be willing to service read and write requests from other nearby machines. In this architecture, portable computers can be servers for some databases and clients for others. The Bayou system is an example that implements the collaborative group architecture.

5.7 Application-Specific Proxy

The Application-specific proxy acts as an intermediary between clients and servers to perform computation-intensive and storage-intensive tasks on behalf of clients. An application-specific proxy on a stationary host supports proxy agents (which can be fixed or mobile) to dynamically "transcended" or "distill" application data to reduce the bandwidth consumption between the proxy and the mobile client. This proxy allows legacy and other nonstandard (including thin) clients to inter-operate with existing servers. From the client's perspective, the proxy is simply a server that gets the data from someplace else. The examples of application-specific proxies are discussed in Brooks et al.Brewer et at and Zexel and Decamp.

5.8 Virtual Mobility of Servers

In a wireless information system, information (data) servers are connected via fixed networks to provide information services to mobile users. The replication (or partition) of information services could help reduce the latency of remote operations and balance the workloads of data servers in a distributed and multiple networks environment. The replication of information services in different networks can be used to support the application service handoffs for moving mobile clients. In a wireless environment, the client moves around, perhaps to areas it has never visited before; once in a new milieu (or a new network environment), it will negotiate with some nearby machine to become a new coordinator to continuously provide services for its operations. The movement of mobile hosts may result in a long path of communications in fixed networks because the physical distance may not necessarily reflect network and service distance between the client and the server, especially when the movement crosses the boundary of different networks. The long path of communication in fixed networks will increase the traffic and latency of transactional services. If the new coordinator could always use a nearby or

local information service site as the client's data server, the traffic and latency in fixed networks can be reduced for the continuous interaction between the client and the information service server. Therefore, the mobility of the client introduces the concepts of virtual mobility of servers and application service handoffs. An issue of supporting the virtual mobility of servers is to minimize the overhead of application service handoffs for synchronous multi-server operations. The work in Taut and Decamp investigates how to maintain replicas in a distributed file system that supports mobile clients. This work assumes that (1) clients' movements cannot be constrained, although patterns of movement may exist; (2) the latency of remote operations increases as the distance between hosts increases; (3) sequential sharing workload is not uncommon, but simultaneous sharing (other than read-read) workload is rare; and (4) a file cache of modest size is maintained by each client. The design goals in this work include: minimizing ACM Computing Surveys, Vol. 31, No. 2, June 1999 synchronous multi-server operations; easing the addition and deletion of server sites; and allowing for incomplete replication. To achieve these goals, primary secondary server hierarchy architecture is proposed, as shown in Figure 5.4. Typically, the client need not contact any (remote) server, but it communicates synchronously with the (local) primary server only. All updates are stored in the client's cache. The primary server makes periodic pickups from the clients it is currently servicing and propagates updates back to the secondary servers asynchronously. This pickup strategy allows the client's write operation to return immediately after placing the new value in its cache. The primary servers are used as intermediaries between clients and secondary servers. Because the system supports replica addition/deletion and incomplete replication, a primary server must learn about mapping from the file system to the secondary servers. The mapping is provided by calling a mapping server function specified by the client. For example, someone from New York who is visiting Seattle would call the Seattle matchmaker to obtain a local primary server site. The locally-chosen primary server then calls the mapping server in New York to locate files that the client wishes to access. After the handoff of primary servers, the new primary server can lazily copy the client's file from the old cache in the old primary server. Therefore, this method always connects the mobile client to the local primary server for information access in a distributed file

system. The work in Krishna Kumar and Jain and Krishna Kumar presents a system architecture and application service handoff protocols for virtual mobility of servers. The architecture is based on replicated distributed servers connected to users via a personal communication services (PCS) network. Under this architecture, as the user moves out of one service area into another, the local server at the new service area takes over providing the service. This service handoff for the virtual mobility of the server is broadly analogous to the PCS call handoff procedure and also requires that service continuation is transparent with no interruptions. For the application services handoffs, a service coordinator, when informed that the user has moved, initiates the set-up of a conference call between the current server, the mobile host, and the new server so that the service can be transparently handed off to the new server. The coordinator can then terminate the call with the old server. Before the new server takes over during a service handoff, it has to know what the mobile user is currently doing with the service, that is., the context of the user with respect to the service. Context information is the information associated with a user and a service so

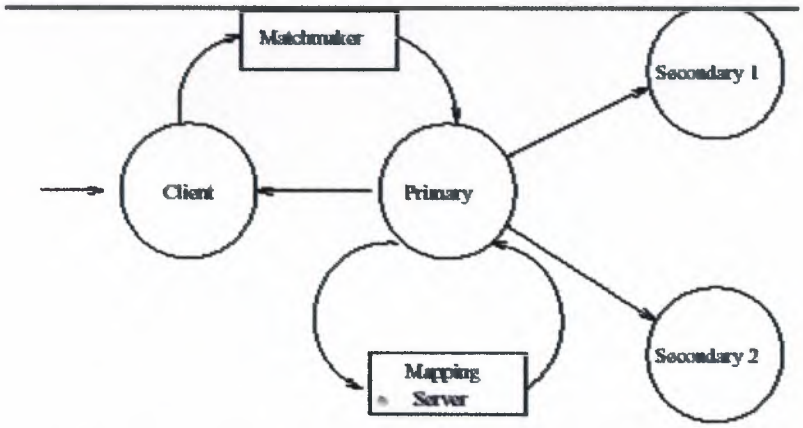

Figure 5.4 The primary secondary server hierarchy

ACM Computing Surveys, the user can access different servers transparently. Part of the context is static, including password and access rights that do not change as the user accesses information. However, the context also includes dynamic session-specific data information such as how much data has been read or modified by the user, whether the changes are meant to be transactional, whether the user holds any locks to access the data, and so on. In Elmagarmid et al. an approach called Reservation Algorithm (RA) is

proposed to avoid transaction log transfer and the use of global commit Protocol.

5.9 Summary

Talking about the architectures models in this chapter we have Info Pad project demonstrates the approach of thin client architecture. The Info Pad system is composed of four layers. Mobile clients must be able to use networks with rather unpleasant characteristics: intermittence, low bandwidth, high latency, or high expense. The connectivity with one or more of these properties is referred to as weak connectivity. Flexible client-server architecture generalizes both thin client and full client architectures in that the roles of clients and servers and application logic can be dynamically relocated and performed on mobile and stationary hosts.

6. MOBILE DATA ACCESS

6.1Overview

This chapter we are going to explain the data communication about the server data, broadcast disks. And client cache management automated hoarding varied granularity of cache coherence and also aches invalidation reports.

6.2Introduction

Mobile data access enables the delivery of server data and the maintenance of client-server data consistency in a mobile and wireless environment. Efficient and consistent data access in mobile environments is a challenge research area because of weak connectivity and resource constraints. The data access strategies in a mobile information system can be characterized by delivery modes, data organizations, and consistency requirements, etc. The mode for server data delivery can be server-push, client-pull, or hybrid. The server-push delivery is initiated by server functions that push data from the server to the clients. The client-pull delivery is initiated by client functions which send requests to a server and "pull" data from the server in order to provide data to locally running applications. The hybrid delivery uses both server-push and client- pull delivery. The data organizations include mobility-specific data organizations like mobile database fragments in the server storage and data multiplexing and indexing in the broadcast disk. The consistency requirements range from weak consistency to strong consistency. Figure 6.1 illustrates the new paradigm of mobile data access for mobile information access. In this section, we will examine various proposed approaches that offer the new paradigm of data access in mobile client- server information systems.

6.3 Server Data Dissemination

In many applications (e.g., Web access), the downstream data volume from servers to clients is much greater than the upstream data volume from clients back to servers. The unbalanced communications are referred to as asymmetrical communications between clients and servers. Application examples of asymmetrical communications in wireless

environments include Hughes' mobile hosts usually have a lower bandwidth cellular or PSTN link while servers at fixed hosts may have relatively high bandwidth broadcast capability.

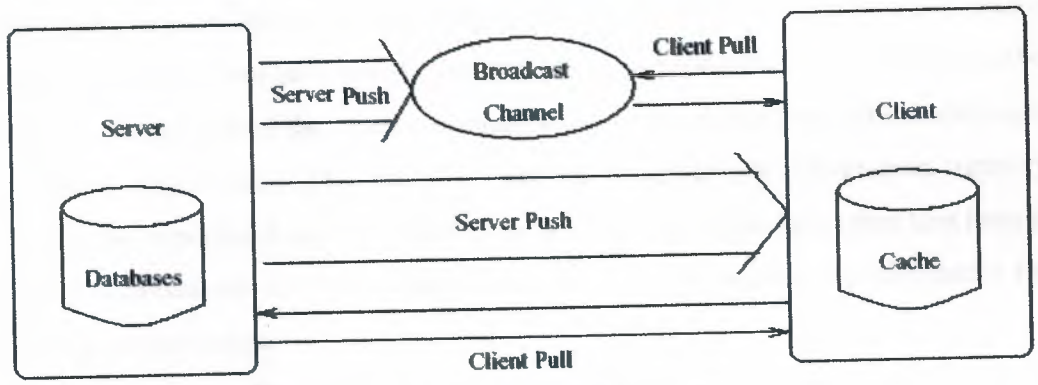

Figure 6.1 A mobile access paradigm

ACM Computing Surveys, placations with asymmetrical communications is how to deliver server data and information to a large number of clients. To address this scalability problem, a new information system architecture that exploits broadcast-based dissemination capability of communications has been proposed. The central idea is that the servers exploit the downstream communication capacity in bandwidth by broadcasting data to multiple clients. This arrangement is called a push-based architecture where data is pushed from the server to the clients. In contrast, most traditional client-server information systems use pull-based data delivery to provide data to locally running applications.

6.4 Broadcast Disks

When a server continuously and repeatedly broadcasts data to a client community, the broadcast channel becomes a "disk" from which clients can retrieve data as it goes by. The broadcasting data can be organized as multiple disks of different sizes and speeds on the broadcast medium .The broadcast is created by multiplexing chunks of data from different disks onto the same broadcast channel. The chunks of each disk are evenly interspersed with each other. The chunks of the fast disks are repeated more often than the chunks of the slow disks (see Figure 6.2). The relative speeds of these disks can be

adjusted as a parameter to the configuration of the broadcast. This use of the channel effectively puts the fast disks closer to the client while at the same time pushing the slower disks further away. This technique presents an opportunity to more closely match the broadcast to the workload at the client. Assuming that the server has an indication of the client access patterns (either by watching their previous activity or from a description of intended future use from each client), then hot pages or pages that are more likely to be of interest to a larger part of the client community can be brought closer while cold pages can be pushed further away. This, in effect, creates an arbitrarily fine-grained memory hierarchy, as the expected delay in obtaining an item depends upon how often that item is broadcast. The broadcast disk technique, therefore, provides improved performance for no uniformly accessed data.

In the simplest scenario, the server can broadcast different items at the same frequency. With the "flat" broadcast, the expected delay required prior to obtaining an item is the same for all items broadcast regardless of their relative importance to the clients. This "flat" approach has been adopted in earlier work on broadcast-based information system such as Data cycle and the work in Imielinski et al By comparison, the server can broadcast different items with differing frequency; important items can be broadcast more often than others.

Client-Server Computing in Mobile Environments

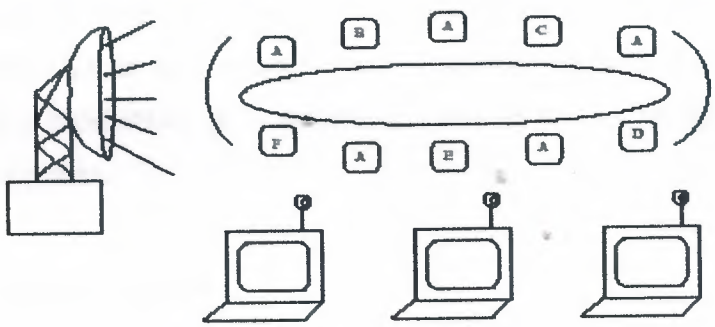

Figure 6.2 A broadcast program

ACM Computing Surveys, Similar to the broadcast disk concept, a pyramid broadcasting method is used to provide Video-On-Demand services to mobile users. In pyramid broadcasting, the most frequently requested movies are multiplexed on the broadcast network, resulting in radical improvement of access time and efficient bandwidth utilization.

6.5 Indexing on Air

In the "push based" approach, servers periodically broadcast most frequently requested data items (hot spots). The server should dynamically adjust the content of the broadcast hot spot, depending on the periodically measured demand distribution. The client is lazy in that it transmits only when necessary. The client could also stay in the doze mode (turning the power off) as much as possible. To minimize wake-up time, clients can use selective tuning capabilities in the broadcasting channel. In Imielinski et al. the authors discuss basic methods for data organization on the broadcast channel that provides selective tuning capabilities for Clients. In these methods, index information is broadcast together with the data. The index is structured so that the following two parameters are minimized: Query Time: Amount of time that it takes for a client to issue a query until the answer is received by the client. Listening Time: Amount of time spent by the client listening to the channel. Query time is proportional to the overall size of the broadcast data. Therefore, the presence of the index increases the query time since the presence of the index increases the overall broadcast size. However, the presence of the index reduces the listening time. This, in turn, reduces energy consumption because clients can use the selective listening capabilities in broadcasting channels to stay in the doze mode and minimize wake-up time.

6.6 Client Cache Management

Caching of frequently-accessed data items is an important technique that reduces contention and improves query response times on narrow bandwidth wireless links. The cached data can also support disconnected or intermitted connected operations. However, cache pre-fetching and consistency strategies can be greatly affected by the disconnection or weak connectivity of mobile clients. The weak connectivity makes cache coherence

expensive due to communication latency and intermittent failures. Pre-fetching (or hoarding) data into the client cache prior to disconnection is a difficult challenge in mobile client-server computing. This subsection describes an automated hoarding approach and two cache validation mechanisms.

6.7 Automated Hoarding.

A useful solution to support disconnected operations is hoarding, in which nonsocial files are cached on the client cache prior to disconnection. The difficult issue for hoarding is which files should be selected and stored locally. Possible solutions include choosing the most recently referenced files or asking the user to participate at least peripherally in managing hoard contents. The former approach might be wasteful of scarce hoard space, while the latter requires more expertise and involvement that most users are willing to offer. In the SEER system, an automated predictive hoarding approach is developed. The automated predictive hoarding is based on the idea that a system can observe user behavior, make inferences about the semantic relationships between files, and use those inferences to aid the user. In SEER, an observer component watches the user's behavior and file accesses, classifying each access according.

ACM Computing Surveys, to type, converting path names to absolute format, and feeding the results to a correlate component. The correlate component evaluates the file references, calculating the semantic distances among various files. These semantic distances drive a clustering algorithm that assigns each file to one or more projects. When new hoard contents are to be chosen, the correlate component examines the projects to find those that are currently active, and selects the highest-priority projects until the maximum hoard size is reached. In this way, SEER is able to operate without user intervention (though the user might involve informing the computer that a disconnection is imminent). The fundamental assumption of SEER is that there is semantic locality in user behavior. By detecting and exploiting this locality, a system can make inferences about the relationships between various files. Once these relationships are known, it is possible to automate the hoarding. To detect semantic locality, SEER uses a concept known as semantic distance. Conceptually, semantic distance attempts to quantify a user's intuition about the relationship between files. A low semantic distance suggests

that the files are closely related and thus are probably involved in the same project, while a large value indicates relative independence and different projects. The semantic distance is based on measurements of individual file references, rather than looking at the files themselves. The distance between references is then summarized to produce a value that is relevant to the individual files. Several semantic distance measurement methods are defined based on file references in Kenning and Pope.

6.8 Varied Granularity of Cache Coherence

Consistency methods in traditional client-server architecture can be divided into two categories: (1) callback approach when servers send invalidation messages directly to the clients that have cached the data items to be updated and (2) detection approach when clients send queries to servers to validate cached data. The difficulty of using these traditional methods in mobile environments is due to the disconnection and weak connectivity of clients. Frequently disconnected clients make it very ineffective to use the detection approach. On the other hand, the classic callback approach may also be very expensive, due to network latency or intermittent failures. After a long disconnection, many data items at the server side may have been updated. In this case, the time for the callback invalidation of each data item can be substantial on a low network. In the Coda system. Clients can track the server state at multiple levels of granularity. A server maintains version stamps for each of its file volumes, in addition to stamps on individual objects (or items). When an object is updated, the server increments the version stamps of the object and that of its containing volume. Clients cache volume version stamps in anticipation of disconnection. When connectivity is restored after a network failure, the client presents volume stamps for validation. If a volume stamp is still valid, then so is every object cached from the volume. If a volume stamp is not valid, cached objects from the volume need to be validated individually. Even in this case, performance is no worse than in the original scheme. On the other hand, if the validation for each individual data item is not possible due to slow connections, the client can assume that all items in the volume are invalid or defer the validation until the time when the item is used. The experiments and measurements from the Coda system confirm that the varied granularity of cache coherence dramatically improve the speed of cache validation.

6.9 Cache Invalidation Reports.

A dissemination-based approach to the problem of invalidating caches in wireless environments is proposed in Barbara and Imielinski by utilizing wireless broadcast channels. In this approach, a server periodically broadcasts an invalidation report that reports data items which have been changed. Rather than querying a server directly regarding the validation of cached copies, clients listen to these invalidation reports over broadcast channels. This approach is attractive because a server does not need to know the location and connection status of its clients, and the clients do not need to establish an "uplink" connection to the server to invalidate their caches. Three algorithms are presented in Barbara and Imielinski namely, the Broadcasting Timestamps (TS), Amnesic Terminals (AT), and Signatures (SIG). In the TS algorithm, the invalidation report includes only the information regarding the data items that have been updated within the preceding *w* seconds. The report includes the names of these items and the timestamps at which they were updated. The invalidation report in the AT algorithm includes the identifiers of data items that were updated during the last broadcast period *L*. In both the TS and AT algorithms, clients must invalidate their entire

Cache when their disconnection period exceeds a specified length (*w* seconds for TS and L seconds for AT). In the SIG algorithm, the report contains a set of combined signatures of data items.2 the structure and size of these signatures are designed to diagnose unto f differing items. If more than f different items (it does not matter whether these items had been cached or not) are updated in the data server since the combined signatures were last cached, most items cached by the clients will be invalidated by the SIG algorithm, although many are in fact valid. An improved invalidation report method called Bits-Sequences (BS) is proposed in Jingo et al. this method adapts to long disconnected clients. In the Bit-Sequences (BS) algorithm, the server broadcasts a set of bit sequences. Each sequence consists of a series of binary bits and is associated with a timestamp. Each bit represents a data item in a database. A bit "1" in a sequence means that the item represented by the bit has been updated since the time specified by the timestamp of the sequence. A bit "0" means that that item has not been updated since that time. Clients must check the invalidation report before they can use their caches for query processing. If a bit sequence among the sequence set with the most recent timestamp equals to or

predates the disconnection time of the client, the sequence will be used to invalidate its caches. The data items represented by these "1" bits in the sequence will be Invalidated. If there is no such sequence (i.e., the disconnection time precedes the timestamp of the highest sequence), the entire cache in the client will be invalidated. Consider a database consisting of 16 data items. Shows Bit-Sequences (BS) structure reported by a server at the time 250. Suppose a client listens to the report after having slept for 80 time units. That is, the client disconnected at the time 170 (5250-80), which is larger than TS~B2! But less than TS~B1! The client will then use B2 to invalidate its caches. To locate the items denoted by the two "1" bits in B2, the client will check both B3 and B4 sequences, using the following proce-2Signatures are checksums computed over the value of data items in the database. A combined signature is the Exclusive OR of a set of individual signatures. Each combined signature, therefore, represents a subset of data items. In the SIG algorithm, m combined signatures are computed such that an item i is in the set of combined signature so with probability 1/~f 1 1! Where f is the number of differing items up to which the algorithm can diagnose. To locate the second bit that is set to "1" in B2, check the position of the second "1" bit in B3. The second "1" bit in B3 is in the 5th position; therefore, check the position of the 5th "1" bit in B4. Because B4 is the highest sequence and the 5th "1" bit in B4 is in the 8^{th} position, the client concludes that the 8th data item was updated since the time 170. Similarly, the client can deduce that the 12th data item has also been updated since that time. Therefore, both the 8th and 12th data items will be invalidated. This method works effectively for clients who have been disconnected for a long time. It optimizes the utilization of bandwidth for invalidation report. In Picture and Samaras a revised version of invalidation reports is designed to provide the semantics of read-only transactions for mobile clients without sending uplink requests to Servers.

6.10 Summary

Talking about data communication in many applications (e.g., Web access), the downstream data volume from servers to clients is much greater than the upstream data volume from clients back to servers. The unbalanced communications are referred to as asymmetrical communications between clients and servers. When a server continuously and repeatedly broadcasts data to a client community, the broadcast channel becomes a "disk" from which clients can retrieve data as it goes by. The broadcasting data can be organized as multiple disks of different sizes and speeds on the broadcast medium. In the "push based" approach, servers periodically broadcast most frequently requested data items. Caching of frequently-accessed data items is an important technique that reduces contention and improves query response times on narrow bandwidth wireless links.

CONCLUSIONS

There are many text entry methods available to designers of mobile systems, and without a doubt more are forthcoming. However, deciding which is best for an application is difficult, in part, because of the lack of publications giving empirically measured text entry speeds and accuracies. This paper has brought together many of the techniques in use or under investigation in this exciting area in mobile computing. The result is a snapshot of the current state of the art in mobile text entry.

Some important modelling techniques have been presented and elaborated. Movement and language are omnipresent in human-computer interaction. We have shown how Fitts' law and a language corpus can work together in a priori analyses of design alternatives for stylus input on soft keyboards or single-finger input on small physical keyboards. However further work is needed to refine this modelling technique, for example, in determining the correct coefficient in the Fitts' law model and in exploring and refining other aspects of the model, such as treatment of the space and punctuation characters, or its sensitivity to changes in the language model.

Additionally, we have examined many issues in methodology and evaluation, and have identified factors, such as focus of attention, and whether one or two hands are used to manipulate the text entry device, that impact user performance. Clearly, evaluation is critical, and it is by no means simple. A number of issues are particularly tricky such as the measurement and treatment of errors and the types of tasks used in text entry studies. These and other topics, such as internationalization, are the subject of ongoing and future work