SOLUTIONS FOR DYNAMIC CHANNEL ASSIGNMENT AND SYNCHRONIZATION PROBLEM FOR WIRELESS MULTIMEDIA SYSTEM

SungBum Hong, M.S.

Dissertation Prepared for the Degree of
DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS
August 2002

APPROVED:
Roy T Jacob, Major Professor
Azzdine Boukerche, Committee Member
Brian O'Connor, Committee Member
Phil Turner, Dean of the School of
Library and Information Science
C. Neal Tate, Dean of the Robert B. Toulouse
School of Graduate Studies

Hong, SungBum, Efficient Solutions for Dynamic Channel Assignment and the Synchronization Problem for Wireless Multimedia System. Doctor of Philosophy (Information Science), Aug. 2002, pp. 90, 4 tables, 48 figures, references, 39 titles.

The recent advances in mobile computing and distributed multimedia systems allow mobile hosts (clients) to access wireless multimedia data anywhere and at anytime. In accessing multimedia information on the distributed multimedia servers from wireless personal communication service systems, a channel assignment problem and synchronization problems should be solved efficiently.

Recent demand for mobile telephone service have been growing rapidly while the electro-magnetic spectrum of frequencies allocated for this purpose remain limited. Any solution to the channel assignment problem is subject to this limitation, as well as the interference constraint between adjacent channels in the spectrum. Channel allocation schemes provide a flexible and efficient access to bandwidth in wireless and mobile communication systems.

In this dissertation, both an efficient distributed algorithm for dynamic channel allocation based upon mutual exclusion model, and an efficient distributed synchronization algorithm using Quasi-sink for wireless and mobile multimedia systems to ensure and facilitate mobile client access to multimedia objects are proposed.

The algorithm's performance with several channel systems using different types of call arrival patterns is determined analytically. A set of simulation experiments to evaluate the performance of our scheme using message complexity and buffer usage at each frame arrival time.

ACKNOWLEDGMENTS

I would like to thank my adviser Dr. Roy T. Jacob for the benefit of his wisdom and expertise which he so generously shared with me. I will be forever grateful for the experience I gained while working with him. I also wish to thank Dr. Azzedine Boukerche for the advice and leadership he has given me. My dissertation is much stronger because of him. I also thank the third member of my committee, Dr. Brian O'Connor, for his many kindnesses, his time and suggestions. Finally, I thank the faculty of the Interdisciplinary Program in Information Science, especially Dr. Samantha K. Hastings, for their instruction.

I dedicate this dissertation to my family. Without their love and support, this dissertation would not be possible.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

During the last decade, the area of *wireless communication* and *mobile computing* has been enormously developed. Wireless communication and mobile computing is taking an important role in increasing the usage of information super-highway. The ultimate goal of this research area allows information seekers to access information at any time at any place. Advanced technologies have created new kinds of information systems. We need to categorize these new systems to ubiquitously access information. The wireless communication using radio such as cellular mobile telephony, PCS (personal communications system) and portable (e.g. laptop, palm-top) computers creates new types of services such as video-phone on PCS and movies on laptops at any place. This technology gives information seekers more freedom from certain places that can offer accessibility of information super-highway.

The concept of cellular architecture as shown in Figure 1.1 is like a collection of hexagonal-shaped geometric areas called *cells*, where each cell is served by an antenna or *base station* (BS). Cell size and shape depend on signal strength and the presence of obstacles to signal propagation. A set of BSs are controlled by a *base station controller* (BSC). Several BSs are connected to a *mobile switching center* (MSC), which manages all calls in a large geographical area, doing call set-up and hand-off. The links between the BSs and MHs are wireless, whereas those between the BSs and their MSCs are wired. This architecture allows a BS to communicate with many MHs concurrently. All of the MHs can move from one BS to another.

Unlike wired computing, mobile computing has extra components: (*i*) wireless communication; (*ii*) mobility; and (*iii*) portability. All problems can be categorized into these three parts but some issues are related with multiple components. The major function of wireless communication is bandwidth management and allocation to mobile clients. The mobility issues include location management, data management,

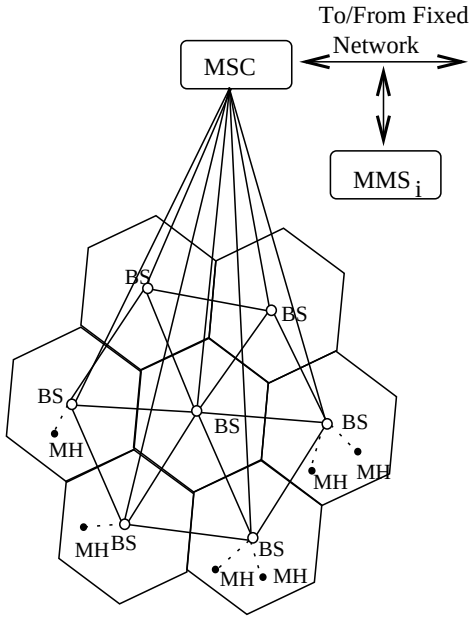

Figure 1.1: Fully Connected Wireless Network

disconnects and transmission security. Portability related with the design of user interface, the design of lightweight terminals with limited power consumption. During the advancements in wireless communication, the rapid development in portable computing platforms, distributed multimedia systems and wireless communication technologies has led to a significant interest in mobile multimedia and wireless networking.

Wireless multimedia systems are widely used in several applications that include VoD (Video-on-Demand), virtual classes for students who can not come to a physical class, virtual stores for people who want to buy goods at home, and multimedia documents. Video entertainment, and multimedia Internet access over wireless networks are some examples of future wireless personal communication systems. The multimedia object has three major characteristics: (*i*) large data that influence storage and communication load; (*ii*) time dependence that also influences delivery time limitation; and (*iii*) media objects that contains audio, video and image which are binary in nature. Each of the characteristics for the wireless distributed multimedia system creates problems. In the first case, depending on services and clients the system can

use a different delivery strategy. For an example, the strategy of VoD for wireless clients splits the video into multiple small size streams and delivers these multiple multimedia units to mobile clients within time limitation. In the second case, each multimedia units has temporal relationship to playout smoothly. In the third case, there also is temporal relationship between audio and video.

In this dissertation, two areas - wireless communication in mobile computing, and time dependent and large data problem in delivering multimedia data are the focal points. In the wireless communication research, an algorithm framework for managing the radio frequency resources are studied. For the time dependent and large data problem, an algorithm framework for delivering multimedia units and a feedback mechanism for managing the buffer of each mobile host are studied.

1.2 Issues and Challenges in Wireless Distributed Multimedia System

1.2.1 Channel Assignment Problem

Even though the growth of mobile communication clients is very rapid, the resources of wireless communication are very limited. The resources are communication channels in the radio time-frequency domain, with users and base stations distributed geographically across the service area. Especially, wireless multimedia clients need the large amount of bandwidth. Some services like e-mail do not really need a large bandwidth like VoD. Unlike fixed communication lines, clients can go almost any where so that demand for number of channels are different according to the areas. Thus, management of the spectrum is necessary to deal with efficient resource allocation among the different services being used by wireless clients.

The main function of resource management is the channel assignment problem. Any solution to the channel assignment problem is subject to this limitation, as well as the interference constraints between adjacent channels in the spectrum. At a high level, the channel assignment problem in mobile communication systems is to maximize the frequency reuse. A user's request to make a call is handled at the nearest base station, which decides whether to accept or to block the call. As a rule, in making that decision the base station communicates with other base stations,

typically using an independent wired signaling network. An accepted call is allocated a channel for communication with its base station, which separately manages the connection (wireless or wired) to the call's destination.

1.2.2 A Multimedia Synchronization Problem for Mobile Hosts

Distributed multimedia system manipulate diverse media objects including text, images, audio, and video. Some of these objects are time dependent on each other when playing, but some of them are not. According to the type of time-dependency, these media objects can be divided into two groups: discrete objects that include text and images [41]; and continuous objects that include audio and video [37].

The basic abstraction for a time constrained media element is a timed stream of media components. The term "media component" means a video frame or audio sample. These components normally must be kept in temporal order when playing them. The process of maintaining this ordering of the media components is called multimedia synchronization [8]. There are two kinds of timing aspects for time constrained elements: (1)intra-media continuity is subject to a real time constraint in handling media elements; (2) inter-media synchronization is subject to temporal correlation during a playback of media elements [44].

One can distinguish the temporal relations according to the use of either live or synthetic synchronization [8]. Live synchronization exactly reproduces the temporal relations made during the capturing process, while synthetic synchronization artificially specifies the temporal relations. Conversational service is one application of live synchronization, while synthetic synchronization is often used in retrieval-based systems to rearrange multimedia objects to produce new combined multimedia objects. Temporal models allow specification of the temporal relations and operations of multimedia objects. These models can express complex operations by combining simple operations such as parallelized media streams, serialized multimedia objects, and simple independent multimedia objects [8].

Even though the synchronization problem for distributed stored multimedia streams based on fixed networks has been well studied [1, 7, 31, 57], little has been done on

wireless networks.

The synchronization problem under a combination of wireless/wired networks is more complicated than the problem for wired networks because wireless networks must use a base station to deliver packets, and have much fewer resources [22, 27]. Furthermore, a mobile/wireless communication system is a wireless network that provides low-power, and high-quality access to *mobile users* or *mobile hosts* (MH).

1.3 Significance of the Study

In this dissertation, efficient solutions for both bandwidth management and multimedia synchronization problem are presented. The contributions for this study are summarized in following sections.

1.3.1 Bandwidth Management

Simple distributed dynamic resource allocation (DDRA) scheme is proposed to cope with the problem of high channel request in managing bandwidth. Without a channel borrowing scheme, DDRA deals with high demand. DDRA algorithm allows any base station to be a host of a group channels and it takes advantage of the fast response time and the low denial rate under a very high system load. As a consequence, performance of DDRA algorithm is analyzed using Queuing model detailed simulation experiments. DDRA algorithm is stable with regard to the different arrival patterns (in term of response time). Comparisons with previous schemes shows significant improvement of performance in high system load. The algorithm runs at each base station (BS) and does not need a Mobile Switch Center (MSC) to control channel usage. The base station in each cell allocates resources available at run time by communicating with its neighboring base stations to exchange information about channel usage. DDRA is a distributed algorithm, and deadlock-free with a limited waiting time, i.e., a Mobile Host (MH) that wants to get permission to set up a communication session will be informed by its base station within a finite time whether a channel is available or not.

1.3.2 Multimedia Synchronization Algorithm

A combination wired/wireless network for multimedia servers differs significantly from a wired network: (*i*) In delivering packets for mobile hosts, all packets must pass through at least one BS to reach a mobile host; (*ii*) a MH has small memory and small screen size; and (*iii*) there is low network bandwidth between a BS and a MH. Because of such differences, conventional synchronization strategies for delivering MMUs to MHs within a time certain can not be applied in the mixed environment. A single base station must pass many MMUs and much control information to MHs, which can cause congestion at the base station. Because of the small memory and low bandwidth at a MH, this traffic may also cause MH memory underflow or overflow. Additionally, current Internet standards do not guarantee on-time packet delivery. All of these problems point to the need for a new synchronization methodology for multimedia applications.

A novel synchronization scheme for wireless clients and distributed multimedia systems is proposed that uses a Quasi-sink to control synchronization. The proposed solution copes with *network jitter*, *end-system jitter*, *clock drift*, and *changing network conditions* [7, 38]. Hence, the scheme is suitable both for synchronizing *inter-* and *intra-streams* during playback video and also for *re-synchronize* streams in a wireless network. A set of simulation experiments to study the performance of the algorithm is presented.

The primary features of the algorithm are: (*i*) It uses a small number of communication packets; (*ii*) It easily copes both with wireless and combination wireless/wired networks; (*iii*) The protocol can compute well-defined starting times; (*iv*) A dynamic play-out schedule can be achieved by adopting various transfer rates; and (*v*) Better buffer management can be created through forecasting. The algorithm assumes that the servers and the network provide sufficient resources to deliver multimedia objects to the mobile hosts. The multimedia data consists of many multimedia objects that may be progressively transmitted over the network. The scheme does not uses global times; when servers generate multimedia objects, they stamp these objects with the

current local time to allow the BSs to calculate round trip delay, jitter and inter-arrival time. A sequence number and a server number are carried by each multimedia units processing time is neglected because it is relatively small. Performance of multimedia synchronization algorithm proposed here is unalloyed using Queuing model detailed simulation experiments. The scheme is stable with regard to the different arrival patterns in term of arrival time within certain range of jitter.

1.4 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 introduces the previous research both the channel assignment problem and the synchronization problem. Chapter 3 presents a description of the algorithms and how they can cope with channel assignment problems and synchronization problems such as start-up, inter and intra-stream synchronization, jitter and resynchronization. The chapter also contains the proof of correctness. In Chapter 4, the scheme is compared with previous and related models and the simulation experiments are reported to evaluate the performance of the algorithm. Chapter 5 details conclusions and speculates on additional area of research.

CHAPTER 2

LITERATURE REVIEW

2.1 Previous Work for Channel Assignment Problem

Mobile computing is a new paradigm of computing, with new communication media, that introduces new issues and some research problems for wireless communication and computing. To support wireless communication, one large area is divided into several small subareas, called cells. A channel refers to the bandwidth of the frequency used to communicate between mobile hosts. The same channel can not be used within a cell to communicate with its neighboring cells, to avoid interference between cells [19]. A basic cellular system consists of a *Mobile Host*(MH), a *Base System* and a *Mobile Switching Center* (MSC). Mobile Hosts have the capability of moving between different locations, while maintaining their connection to the network via a cellular connection. The base station in a cell is a non-movable host with the capability of wireless communication with the MHs within that cell. A Mobile Switching Center, which manages all the calls in a large geographical area, does call set-up and hand-off [36]. The issues being researched have been divided into three groups; one is wireless communication, the second is mobility, and the third is portability. Wireless Communication issues consist of the Cellular Concept and Architecture, Frequency and Channel Assignment Problems, the Hand-off Problem, the Bandwidth problem, and Multiple Access Protocols and Techniques. Mobility Issues can be divided into several problems such as the System Model, Data Management, Location Management, Disconnection and Mobile File systems, Mobile Networking, Mobile RPC, Mobile IP and Security issues. Portability issues include several small problems such as Terminal Characteristics, Low Power Design, User Interface, and Storage Devices [22]. There are two strategies used to assign frequencies to cells: one is fixed channel assignment; the other is dynamic channel assignment. In fixed channel assignment, a fixed number of channels are assigned permanently to each cell. If a MH tries to make a call within a cell, the cell to which the MH belongs will be only served by

the unused channels reserved for that particular cell. If there is no channel available for a new caller in that cell, the call will be blocked or will be made by borrowing a channel from a neighboring cell. Even though there is a shortage of channels in some of the cells, normally there still are numerous unused channels in other cells. In the dynamic strategy, there is no channel assigned permanently to a cell. Each time a call request is made in a cell, the base station in that cell requests a channel from the MSC. According to the algorithm used, which considers the usage of the channel within that cell and its neighboring cells, the MSC allocates channels to requesting cells [36, pages 268–277].

There are several kinds of interference which should be avoided in assigning a channel to a cell. First, there is interference between cells because they are using the same channel. This is called co-channel interference. Second, there is adjacent interference that results from signals which are adjacent in frequency to the desired signal. Avoiding these kinds of interference is crucial to the quality of wireless communications.[36].

Several algorithms for increasing channel utilization have been proposed [2, 11, 13, 14, 16, 17, 36, 40]. Basically, they are classified into two categories: (1) fixed channel assignment (FCA), where channels are permanently allocated to each cell; and (2) dynamic channel assignment (DCA), where all available channels are dynamically allocated upon request. DCA schemes have proven to be more flexible in varying traffic loads than FCA schemes. There are two types of design methodologies in dynamic channel allocation algorithms such as fully dynamic [10, 14] and partly dynamic channel allocation approaches [16, 17, 45]. The fully dynamic channel allocation approach does not require channels transfer mechanism while the other scheme always transfer channels from neighbors under high system load. This study presents a distributed dynamic resource (channel) allocation algorithm (DDRA) for mobile communication systems. The channel assignment model is based upon the mutual exclusion paradigm in distributed systems, using a simple competitive scheme with multiple critical sections [1].

There are a number of different resource allocation problems that are important

[1]Critical section is certain sections of code that can not be executed by more than one process simultaneously

because they are examples for a large class of concurrency control problems. The most classical problems are the dining philosophers [13, 18, 39, 48], the drinking philosophers [12, 54] and the dynamic resource allocation problem [4, 6, 13, 55]. These problems reflect different aspects of the channel allocation problem in wireless communication networks.

In the dining philosophers problem, the process set is fixed and each process uses a fixed set of resources. In the drinking philosophers problem, the process set is fixed, but each process can request a different subset of its maximum resource requirement each time. In the dynamic resource allocation problem, the process set may dynamically change over time, and each process may need different sets of resources at different times.

In [18], Dijkstra initiated the first model for resource allocation,the dining philosophers problem, consisting of a ring of 5 processes, where each process shares a resource with a neighbor. In [39], the problem was generalized to an arbitrary conflict graph where a node represents a process and an edge represents a sharing of resources between two processes. In [39], the shared resources were assigned a partial order that dictates the order in which each process requests the resources it needs. Response time and message complexity are calculated to evaluate algorithm efficiency. Extensive work [13, 43, 48] has been done to improve message complexity and responses time of Lynch's algorithm. The drinking philosophers problem and its solution are introduced [12]. By using the dining philosophers solution as a subroutine, the drinking philosophers problem is solved. The generalized version of the drinking philosophers problem was studied in [54], showing that the solution for drinking philosophers problem can be solved by using any dining philosophers algorithm.

The dynamic resource allocation problem was defined by [4]. It was solved on the assumption that processes must know a priori the IDs of their conflicting processes. In [55], the dynamic resource allocation problem is solved using the solution proposed by [12] as a subroutine. In [6], a synchronous algorithm in a synchronous network and in [14], a dynamic resource allocation algorithm for worst case response time and time complexity are studied.

In the channel assignment problem, the same resource (*i.e.*, a channel) can be

used at different locations. Hence, this problem is much more concurrent than the dynamic resource allocation problem. According to the previous work, the model of the dynamic resource allocation problem in this research is similar to [4] but more general in that it does not assume a priori information about neighboring processes. The resource allocation problem in distributed systems has received quite a lot of attention in the past decade. However, in these many formulations of the resource allocation problem, the problem of finding an optimal allocation is found to be NP-hard [24] in all but very restricted cases. These algorithms are not suitable for channel allocation in mobile systems since a resource can be used in a cluster as long as that use does not interfere with its use in another cluster.

2.2 Previous Work for Mobile Synchronization Problem

Research to synchronize play-out and delivery of distributed stored multimedia streams has included several areas, such as scheduling for synchronization, feedback techniques for synchronization, network-based schemes and buffering-based schemes. To achieve a better solution for synchronization and smooth play-out, this research has concentrated on five of these sub-areas. Existing wireless multimedia synchronization scheme are reviewed.

We can divide the area into two groups: (*i*) a temporal abstraction model providing illustrative tools for defining synchronization semantics for delivery and play-out multimedia objects [15, 44, 46]; (*ii*)synchronization enforcement over the various layers, including operating system, networking, and application layers. In this dissertation, synchronized delivery and play-out issues are focused.

- **Temporal Models**

  Many research attempts have been made to develop conceptual models for representing multimedia components. To specify the temporal behavior of multimedia components, several models are proposed such as an hierarchical model, Petri-Net Models, and a temporal reference framework. First, a hierarchical

model for specifying multimedia components is proposed in [15]. The purpose of the model is to specify and validate multimedia components. Most of the components are compound components so that the hierarchical nature of compound components proposes a hierarchical model and hierarchical design procedures. Second, an example of Petri-net models [46] is Object Composition Petri-net (OCPN) which allows users to randomly access various objects while users are browsing through multimedia information. The description of all the temporal relationships among objects is a major part of a specification model. The OCPN is not directly executable in the sense that it does not specify communication and synchronization over a network in a distributed environment. The dissertation proposes a scheme for multimedia object communication that allows for inter-media synchronization among the multiple streams. To overcome the disadvantage of OCPN, a Petri-net model called eXtended Object composition Petri-net (XOCPN) [46] is proposed. it allows users to describe the communication, transmission and synchronization operation. Third, the paper [44] shows a uniform, theoretical foundation for discussing multimedia synchronization and temporal specification. A temporal reference framework allows users to compare existing temporal specification schemes and their relationships to multimedia synchronization.

- **Scheduling for synchronization**

  According to the location of the scheduler of synchronization, synchronization schedulers are divided into two groups, distributed and centralized. Distributed schedulers can be divided into several subgroups: sender-based, receiver-based, and both-side-based schedulers. A complete software control architecture [34] was suggested by Lamont *et al* that delivers streams in a centralized manner based on traffic prediction and documents specification, Distributed synchronization schedulers [30, 31] have also been proposed, in which a main function of the scheduler is to determine optimal scheduling times for retrieval of multimedia objects from distributed multimedia servers [30]. Their algorithm [31] can adapt quality-of-service parameters dynamically.

- **Feedback techniques for synchronization**

  Distributed multimedia environments allow a multimedia server to service multiple clients who want to play-out multimedia objects. These remote users want to play-out the multimedia objects at the same time and also want to get their multimedia objects without any interruption. The network may have jitter [2] delays, and the multimedia objects in play-out and recording may have non-deterministic variations. To maintain intra-stream synchronization requires a minimum buffer size that is the sum of the two buffers needed to compensate for network jitter and to counteract the variations in play-out rates. Without globally-synchronized clocks, a feedback technique [47] can be used to make synchronized intra-stream and inter-stream multimedia objects correctly play out over networks.

  In the feedback technique [47], intra-stream synchronization allows continuous play-out of multimedia objects at remote clients. Feedback messages are sent back to the multimedia server by the remote clients and are used to evaluate the play-out of multimedia objects at the remote stations. The server uses this result to detect buffer overflow or underflow at remote client sites. Based on this feedback information, transmission rates will be changed to avoid data loss. As soon as the server gets a feedback message, it estimates the difference between the earliest and latest playback times of the synchronization interval units. One protocol [7] uses the feedback messages to compute the buffering required to achieve both continuity between streams and synchronization within the streams. A feedback technique [38] uses a kind of buffer-level control and a nominal buffer range to synchronize inter-streams. To re-synchronize inter-streams, the feedback technique duplicates and drops media units.

[2] *Jitter* is the difference between service times experienced by any two packets

- **Network-based schemes**

  A *network-based scheme* [20] for a flow synchronization protocol can be deployed in distributed multimedia systems. The scheme handles both network delay and the synchronization of multiple flows, but requires globally-synchronized clocks. One scheme uses synchronization [57] to support on-time delivery of packets, jitter control and network resource allocation. To build effective network support, Znati *at al* suggest three basic design requirements: a user interface for network resource requests, provisions guaranteeing the performance of network resources, and requirements for utilizing global network resources. The paper [26] suggests how to manage the buffers at switches and classify packet scheduling algorithms into two classes that are Guaranteed Rate scheduling algorithms (GR) and Rate Controlled Service Disciplines (RCSD). According to the experiment, increasing buffers at switches of RCSD servers can not get higher achievable utilization of network. In [56], spare disk bandwidth and buffer are studied to maximize the system throughput of multimedia servers. To utilize spare system resources, a criteria introduced in [56] to select proper size of multimedia objects and the criteria makes buffer consumption minimized. This leads to maximize system throughput.

- **Schemes for Wireless Network**

  Most of wireless multimedia services still need more network resources than current available wireless network. Depend on resource sufficiency for wireless networks, access speed and mobility are vary so that synchronization schemes are differently developed to fit for each network condition. Here, synchronization schemes for several wireless network type are reviewed according to the types.

  - *Wireless LAN*: Wireless LANs provide wide bandwidth within geographical areas and support low speed mobility. A couple of papers [49, 50, 51], were produced for this research. In [50], four lip synchronization schemes are studied with *single transport streams*. In [51], a set of lip synchronization schemes have been studied with single- stream approach, and the schemes are implemented on wireless LAN (local area network) to measure

the performance of them. In the papers *a single server* is used to store multimedia units for all models.

- *PHS (Personal Handy Phone System)*:It is a kinds of micro cellular digital cordless telephone system and covers little more wide geographical area with medium mobility than Wireless LAN but it covers much less area then PCS (Personal Communication System). In the paper[32, 33], applications of slide control scheme have been proposed. The paper [32], has shown the slide control scheme is effective when audio and video streams are transmitted over two wireless channels for mobile hosts. The paper[33] studies the interleaved transmission of audio and video for wireless multimedia synchronization and also studied QoS (Quality of Service) control scheme.
- *Bluetooth Link*: Bluetooth is technology to links mobile terminals and their peripheral equipments over the ISM (Industrial, Scientific and Medical) band. Using this technology, synchronization scheme has been studied and also interference from other system has been done [42].

Unlike multimedia system on the fixed network, wireless network multimedia system for many communication types with different number of streams has been researched. Until now, much research work on wireless multimedia systems has been done, using different model the schemes above can be categorized into peer-to-peer. The schemes developed for wireless multimedia systems are using the existing OSI/ISO Model by adding or modifying one of the layers.

2.3 Summary

In this chapter, literatures for resource allocation problems and multimedia synchronization problem are reviewed. For channel allocation problem, this dissertation introduces different type of solutions and for multimedia synchronization problem, many type of schemes and different system models are introduced. Some specific drawbacks are also mentioned.

CHAPTER 3

RESOURCE ALLOCATION ALGORITHM

Basically, resource allocation algorithms are classified into two categories: (1) fixed channel assignment (FCA), where channels are permanently allocated to each cell; and (2) dynamic channel assignment (DCA), where all available channels are dynamically allocated upon request. DCA schemes have proven to be more flexible in varying traffic loads than FCA schemes. There are two types of design methodologies in dynamic channel allocation algorithms such as fully dynamic and partly dynamic channel allocation approaches. The fully dynamic dynamic channel allocation approach dose not require channel transfer mechanism while the other does always transferring channel from neighbors under high system load. In this paper, a distributed dynamic resource (channel) allocation algorithm (DDRA) is presented with fully dynamic assigning methodology for mobile communication systems.

3.1 System Model for Channel Assignment Problem

The channel assignment model is based upon the mutual exclusion paradigm in distributed systems using a simple competitive scheme. The following model is adopted. There is a one-one correspondence between processes in the mobile network and vertices in an interference graph (V, E). A process p_i in the network model represents a base station B_i in the graph (V, E), while the edges represent channels. If there are M channels in the wireless communication system and the channels are divided into three groups [1], then there are $M/3$ channels in each group. In this case, three edges are needed between adjacent vertices, since the processes at neighboring vertices must have exclusive access to a channel group. By the three-coloring theorem [53], if a unique index to each group is designed, then group indices may be assigned to the vertices so that two adjacent vertices have different indices. Hence, all vertices can concurrently hold groups while maintaining the mutual exclusion property.

[1] "group" is used as a critical section so that "holding a group" means "entering critical section"

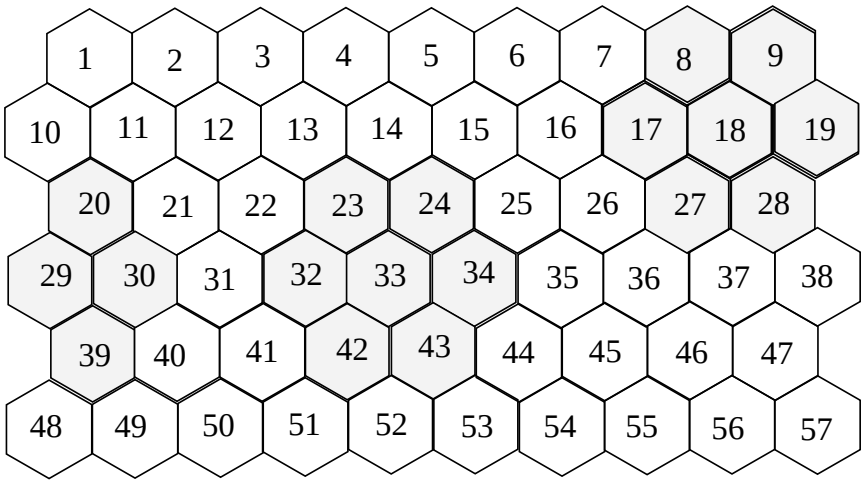

Figure 3.1: A Cellular System

If a channel being used in a cell C_i, then the channel may also be used in any cell C_j which is far enough from the cell C_i to avoid co-channel interference. The *channel reuse* refers to the reuse of a channel to cover different cells which is not close to enough to interfere each other. All channels are orthogonal. Thus, the problem of the adjacent channel interference is not considered. In the model, a 3-cell cluster system is used, *i.e.*, a channel may be used concurrently in cells that are two hops away from each other. For example, in Figure 3.1, the same channel can be used in both cells c_{32} and c_{34} concurrently. In Figure 3.1, the cells c_{23}, c_{24}, c_{32} c_{34}, c_{42}, and c_{43} are neighbors of c_{33}, (its *neighborhood).* Within such a neighborhood, a group can be assigned to two cells at the same time as long as those cells are not neighbors of each other. However, within a cluster of three cells all of whom are neighbors of the others, a group can not be assigned to two of them concurrently.

The following two types of channels are used: (1) *control* channels; and (2) *communication* channels. Control channels consist of *forward* set-up channels and *reverse* set-up channels. While forward set-up channels carry data from MHs to BSs, the reverse set-up channels carry data from BSs to MHs. To initiate a communication session with MH_j, MH_i contacts its base station (B_i), which then connects to the appropriate base station(B_j) through the wired network [5]. The base station(B_j)

then initiates the communication session with MH_j located in same cell as B_j. Both base stations try to find free channels in their cell. If there is no free channel in one of the cells, then a communication session will not be set-up. If both MH_j and MH_i are located in the same cell as the base station (B_j), then B_j tries to find two free channels. After the communication is terminated by either one of the mobile hosts, both channels can be freed to be used by other mobile hosts.

3.2 A Distributed Dynamic Channel Allocation Algorithm

In this section, the distributed dynamic resource/channel allocation algorithm (DDRA) is described. Unlike previous dynamic channel assignment algorithms [14, 40, 45], DDRA divides the channels into equal sized groups where the number of groups equals the number of the stations in a cluster. In Figure 3.2, Each group can be held by a base station at any time unless a neighbor is already holding it. Since each base station makes its decisions based on messages received from all of its neighbors, there is never any need to transfer any channel to another base station.

Each base station can try at any time to get a free channel group that is not held by one of its neighbors. Because each channel group is protected by a critical section, a group can be held by only one base station. The base station can obtain the group only by getting permission from all of its neighbors. Serially obtaining such permissions, a base station visits the channel groups until it finds a free channel in some group. All cells in the wireless communication system can be searching for a free channel concurrently.

Each cell C_i has a simple two dimensional table that contains the information about channel usage for itself and its neighbors. The size of the table is determined by the number of channels in the wireless communication system and the number of cells in a neighborhood. In the model a neighborhood consists of 7 cells, a channel-requesting cell and its neighbors. For example, in Figure 3.1, if cell 33 is using channel Ch_f, then the neighboring cells 23, 24, 32, 34, 42, and 43 can not use Ch_f. Most mobile systems have 666 channels, including 42 control channels, so that there are 624 voice channels [36]. In this case, N, the size of the table in cell C_i, is about $N = 624 * k$,

where k is Number of cells in a cluster bits. To run the DDRA algorithm, a base station needs to maintain only the table and several variables. Only a partition of the table can be accessed by a base station.

The DDRA algorithm runs on all base stations. Each base station has been assigned a unique id number and uses a variable *competition* to count the number of competitions with other base stations required to get a free channel group g_j. One base station in each competition will be the winner, getting g_j. The loser will increment *competition*. This variable will be used to decide the winner of the next competition. When the two competitors have the same *competition* number, then the base station ids are used to select a leader which randomly chooses a winner from those competitors. This scheme makes base stations loosely follow the FCFS rule.

Channel selection is performed according to a *two-step* process. First, a base station must obtain both a *free* channel group and a *permission* from its neighbors to use that channel group shown in Figure 3.2, 3.3, 3.4. Once the base station gets permissions from its neighbors, the ownership of the channel group is used to determine a winner from the competitors. Note that the base stations do several jobs such as acquiring/releasing channels, responding to peer requests, and updating the channel allocation table.

Messages of two types are used in the algorithm, *Request*, and *Reply*. There are two types of request messages that a base station might send: (1) *Request* a free group from its neighbors; and (2) *block* a channel and make a channel group free. There are several type of reply messages that a base station might send: (1) *Reject* a request from another base station; (2) *Free* blocked resource such as a channel or channel group; and (3) *Agree* to allow the requester to use the resource requested.

Now, how the request/reply messages are used is described. Suppose that MH_i requests a free channel from the base station B_i. B_i searches for a free group by picking up a group g_j it has not yet visited and sending a *request* message to all of its neighboring base stations.

Upon receipt of a permission from all its neighbors, B_i searches g_j for a free channel Ch_f. If there is no free channel, B_i keeps trying group after group until it has found a free channel or visited all the available groups.

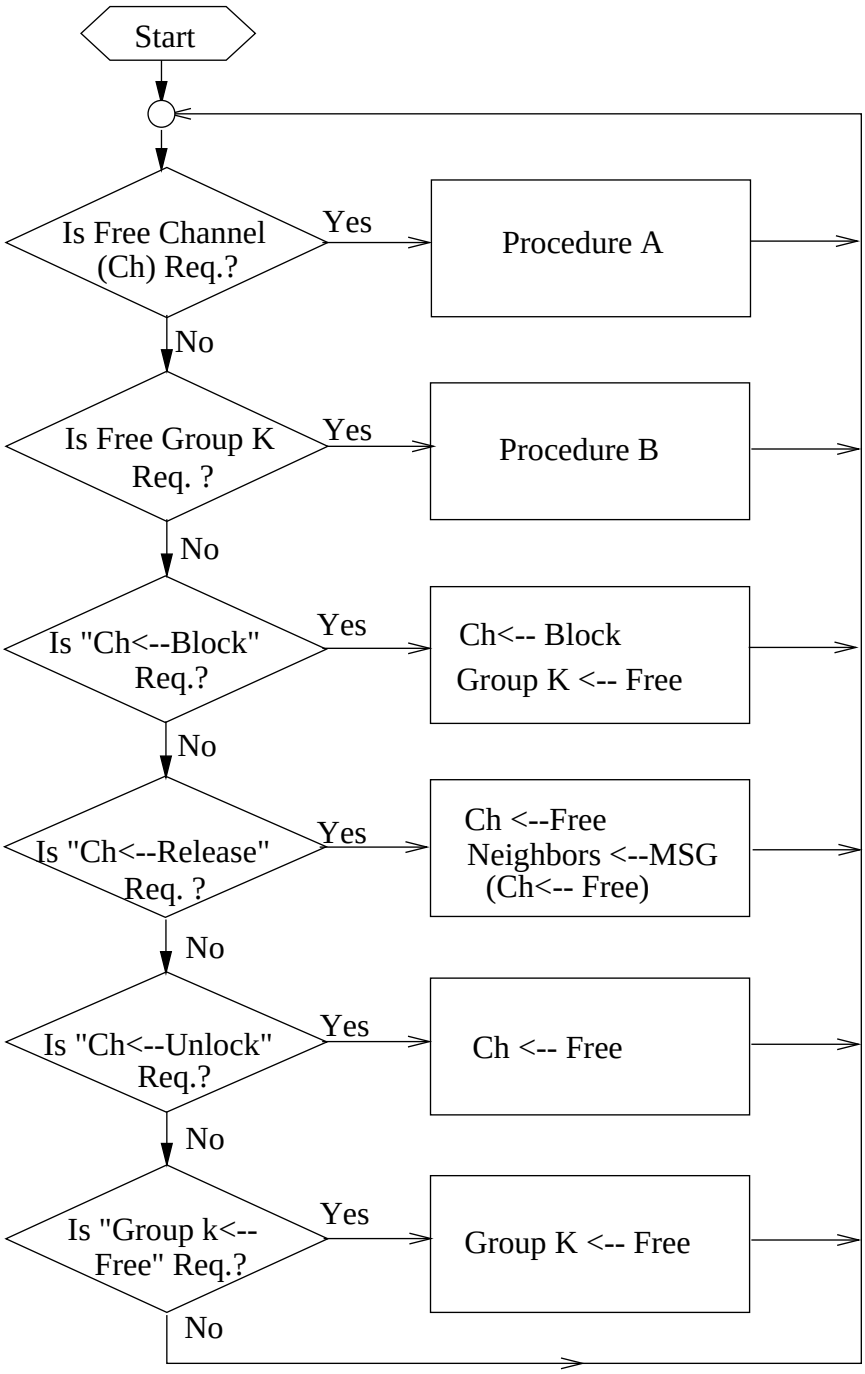

Figure 3.2: DDRA Algorithm

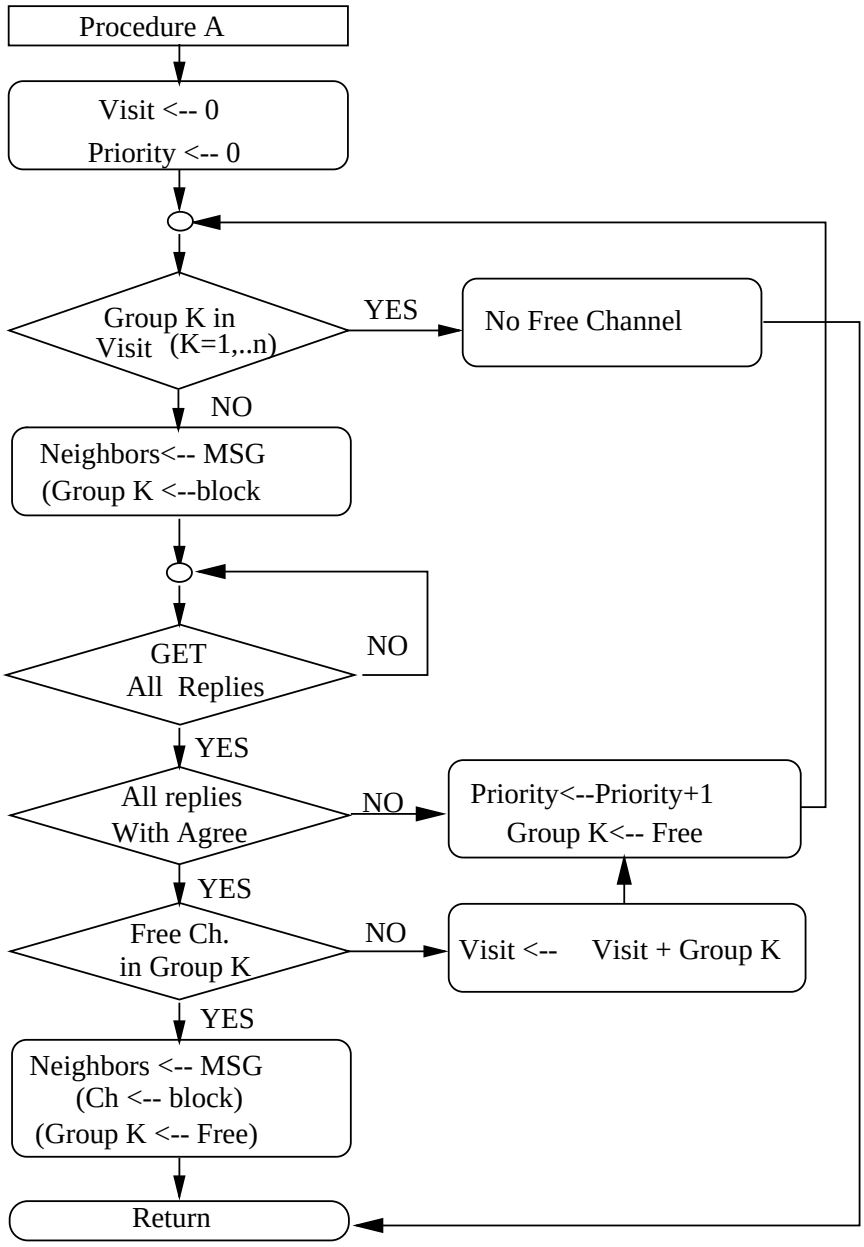

Figure 3.3: Procedure A of the DDRA algorithm

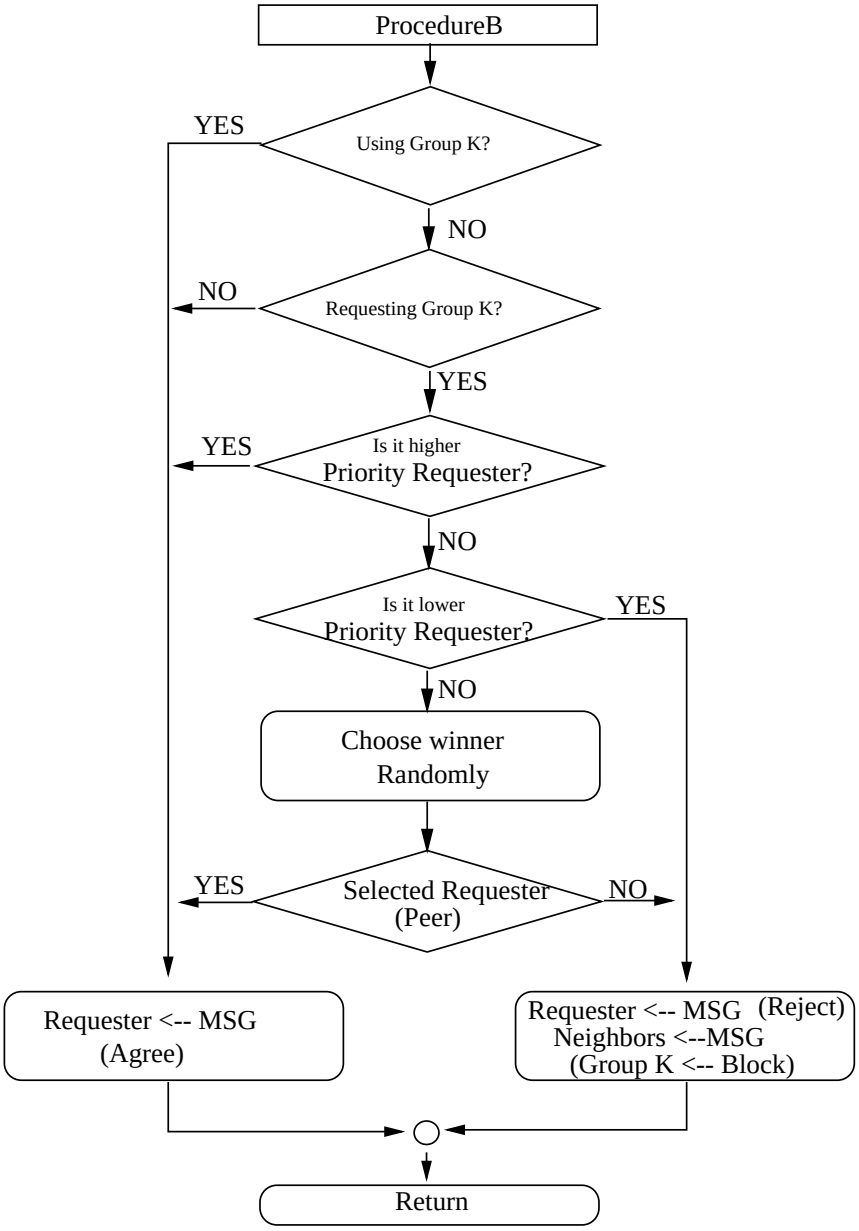

Figure 3.4: Procedure B of the DDRA algorithm

When B_j receives a *request* message from its neighbor B_i, if B_j is using the group g_j, then it sends a *reject* message to B_i. If a base station B_k that is a neighbor of both B_i and B_j is using g_j then B_j sends a *reject* message to B_i. If neither B_j nor B_k is using g_j, B_j sends *agree* to B_i.
If B_j has already requested the group g_j, and receives a *request* message from B_i, then B_j compares their *competition* numbers. Based on the result of the comparison, the winner can be decided, with both sides knowing the identity of the winner. If the *competition* numbers are the same, then both BSs select a chooser according to the their ids, with the station with the largest id being selected. The chooser randomly selects the winner, who claims the free group and resets its own *competition* number to zero. The loser's competition number is increased by one, giving it a greater advantage in the next round. Upon receipt of a *block* message for Ch_f, B_j records the channel Ch_f as blocked in its record.
Upon receipt of a *free* message for Ch_f, B_j marks Ch_f free on its record.
Upon receipt of a release message from MH_i, B_i broadcasts a free message to all of its neighbors.
When B_j gets a *free* message for g_j with a *block* message or Ch_f, it updates its group usage table to reflect the new state.

In paper [10], the formal algorithm is described for the base station B_i by means of pseudo-code, and also the proof of correctness of the algorithm.

3.2.1 An Illustrative Example of DDRA algorithm

In the section, a simple example for the mobile communication model described in section 3.2 is provided to illustrate the features of the algorithm. In the example the following is assumed:

- Cells C_i and C_j are neighbors.
- Base Station B_i is located at C_i and Base Station B_j is located at C_j.
- There are two callers, MH_i at C_i and MH_j at C_j.
- BSs are trying to obtain the free group g_j, and have the same *competition* number.
- The id number of B_j is greater than that of B_i.

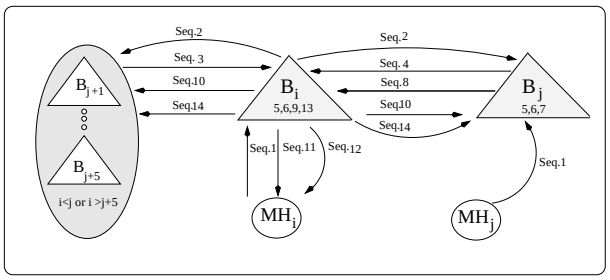

Figure 3.5: Steps in Execution

- There is a free channel $Ch_f \in g_j$.
- A message consists of a tuple <Mes. Type,Source, Req. Group #, Comp. #, Dest.>.

A possible execution sequence of message is given below

Execution Sequence: (check Figure 3.5)

[Seq.1] MH_i requests a free channel from B_i.

[Seq.2] B_i sends a packet $<$ "Req", $B_i, g_j, competition(B_i), B_j >$ to its neighbors.

[Seq.3] All neighbors except B_j reply $<$ "$Agree$", B_i, g_j,0,any BS except $B_j >$ to B_i.

[Seq.4] B_j replies $<$ "Req", $B_j, g_j, competition(B_j), B_i >$.

[Seq.5] Both BSs compares *competition* numbers.

[Seq.6] Since these are same, the BSs compare id numbers.

[Seq.7] B_j randomly selects the winner from B_i and B_j. let B_i be the winner.

[Seq.8] B_i receives a packet $<$ "$agree$", $B_j, g_j, 0, B_i >$ to B_j.

[Seq.9] B_i searches for a free channel Ch_f from g_j.

[Seq.10] B_i sends a packet <"block" Ch_f and free g_j, B_i, g_j, Ch_f,all $B_i'si$ neighbors>.

[Seq.11] B_i assigns the channel Ch_f to MH_i.

[Seq.12] MH_i finishes its session and releases Ch_f.

[Seq.13] The channel Ch_f is free at C_i.

[Seq.14] B_i sends a message $<$ "$free$", $B_i, g_j, Ch_f, all \;\; B_i's \;\; neighbors >$.

3.3 Formal Description of the DDRA Algorithm

We now formally describe the algorithm for a base station B_i by means of pseudo-code:

Case A: Mobile Host MH_i requests a free channel Ch_f from B_i.

Step A0) B_i initializes the group usage table;
and sets *competition*(B_i)$\leftarrow 0$

Step A1) From its group usage table,
B_i selects a group g_j which has not yet been visited

Step A2) **If** B_i has visited all groups **then**
send "wireless network is busy" to MH_i
go to Step 9.

Step A3) $g_j \leftarrow$ block

Step A4) B_i requests the group g_j from all of its neighbors

Step A5) **Wait until** reply from all of its neighbors

Step A6) **If** B_i can not get a free group g_j **then**
competition(B_i) $\leftarrow$ competition(B_i) + 1
$g_j \leftarrow$ Free
go to Step 1.

Step A7) B_i searches for a free channel Ch_f in g_j.

Step A8) **If** B_i finds $Ch_f \in g_j$ **then**
$Ch_f \leftarrow$ block ; $g_j \leftarrow$ Free
B_i sends the message "make g_j free"
and "block Ch_f" to all of its neighbors.
competition(B_i) $\leftarrow 0$
else $g_j \leftarrow$ Free ; group usage table(g_i) $\leftarrow$ Visited
go to step 1.

Step A9) **Exit**.

Case B: B_i receives a request from B_j.

Step B1) **If** the requested g_j is being used at B_i **then**
reply "reject".

Step B2)**If** g_j is Free **then**
reply "agree".
Change the owner of g_j to "others"

Step B4) **If** the requested g_j is also requested by B_i **then**

If *competition*(B_j) > *competition*(B_i)

Reply "agree" to the base stationB_j

$B_i \leftarrow$ "False" to make "fail" for B_i

else If *competition*(B_j) < *competition*(B_i)

Reply "reject" to the base stationB_j

$B_i \leftarrow$ "agree" to make "success" for B_i

else If competition (B_i) == competition(B_j) then

Reply "agree" to the base stationB_j

$B_i \leftarrow$ "False" to make "fail" for B_i

else If *competition*(B_j) < *competition*(B_i)

Reply "reject" to the base stationB_j

$B_i \leftarrow$ "agree" to make "success" for B_i

else If competition (B_i) == competition(B_j) then

If priority(B_j) > priority(B_i)

Randomly select the winner for g_j

If B_i was selected then

Send "reject" to base stationB_j

$B_i \leftarrow$ "agree"

else

Send "agree" to base stationB_j

$B_i \leftarrow$ "False"

Case C: B_i receives a message for "block Ch_f" and "free g_i".

Step C1) Channel $Ch_f \leftarrow$ "block"

Step C2) group $g_i \leftarrow$ "Free"

Case D: B_i receives a "Release Ch_f" message from MH_k.

Step D1) Channel $Ch_f \leftarrow$ "Free"

Step D2) Broadcast "Ch_f is Free" to all of B_i's neighbors

Case E: B_i receives a "Free Ch_f" message from B_j.

Step E1) Channel $Ch_f \leftarrow$ "Free"

Case F: B_i receives a "Free g_i" message from B_j.

Step F1) group $g_i \leftarrow$ "Free"

3.4 Proof of Correctness for DDRA Algorithm

Lemma 1 *The DDRA algorithm always returns a free channel group if there is a group which is not visited by a base station B_i to search for a free channel.*

Proof: Let us assume that the DDRA algorithm can not return a free channel group even though there is a group g_j which has not been visited by B_i. Let N_g denotes the number of groups and also let N_n denotes the number of neighbors. Each time B_i tries without success to get g_j, B_i's variable *competition* is incremented by one see step A6 of DDRA's formal description. When B_i obtains the highest possible *competition* number, i.e., $(N_g - 1)logN_n$, then B_i will always win the next competition for g_j, see step B4 of the DDRA algorithm. Therefore, B_i will eventually gets g_j. This is a contradiction. Therefore, the assumption is false and the lemma is proved.

Lemma 2 *All loops in Case A of the DDRA algorithm are finite and thus will eventually terminate.*

Proof: There are two loops in the DDRA algorithm. One is the inner loop between step A1 and step A6 and the other is the outer loop between step A1 and step A8.

According to Lemma 1, the inner loop of DDRA algorithm always returns a free channel group if there is any group which is not yet visited. If all groups have been visited, step A2 allows the process to be terminated. Hence, either the inner loop finds a free group or terminates in step A2.

In the outer loop whether a free channel is found or not the computation returns to step A8. If a free channel is found, the outer loop is terminated. Since the number of groups is finite, the outer loop can return to A1 to select another free group only finitely many times. In either case, the loop eventually terminates.

Theorem 1 *The DDRA algorithm satisfies the mutual exclusion property.*

Proof: We assume that two neighboring base stations B_i and B_j can simultaneously acquire a channel Ch_f belonging to group g_j. We can divide the send/receive operations of the DDRA algorithm into send/receive operations as follow:

- Choose_group(B_i, g_j) - B_i selects a free group g_j from its table, step A1.
- send_req(B_i, g_j) - B_i requests the group g_j from its neighbors, step A4.
- send_reply(B_i, g_j) - B_i replies to the requester, step B1 to step B4.
- receive_req(B_j, g_j) - B_i receives a request from B_j, step B1 to step B4.
- receive_reply(B_j, g_j) - B_i receives a reply from B_j, step A5.
- ch_acq(B_i, g_j) - B_i acquired Ch_f and send the information to its neighbors, Case C.

The following relationships clearly hold between these events at any base station[2].

At base station B_i:
Choose_group(B_i, g_j) < send_req(B_i, g_j) < receive_reply(B_j, g_j) < ch_acq(B_i, Ch_f).
At base station B_j:
Choose_group(B_j, g_j) < send_req(B_j, g_j) < receive_reply(B_i, g_j) < ch_acq(B_j, ch_f).

To acquire the channel Ch_f, B_i and B_j both must try to acquire g_j since only a base station that has the group g_j can use the channel Ch_f. Furthermore, any base station that wants to use the channel needs to obtain permission from all of its neighbors. If a free channel is to be chosen from g_j by both base stations, g_j must be held concurrently by them. Therefore, the following cases are considered:

Case 1: B_i gets a "receive_req(B_j, g_j)" message from
B_j before it blocks g_j.
In this case, B_j will get the g_j.
Case 2: B_i gets a "receive_req(B_j, g_j)" message from
B_j after it blocks g_j.

[2]Here, the causal order between messages is used; i.e., "$\alpha < \beta$" iff "α happens before β" [35].

We need to split this case follows.

Case 2a: At B_i, "receive_reply(B_j, g_j)" < "receive_req(B_j, g_j)".

In this case, B_i will get the g_j.

Case 2b: At B_i, "receive_req(B_j, g_j)" < "receive_reply(B_j, g_j)".

In the DDRA Algorithm, The owner of g_j is decided by a "competition", so that there is one owner at this moment.

This list of cases is exhaustive, though it guarantees that there is only one owner for one channel group at any time. Thus, the group g_j belongs to exactly one base station at any time. Furthermore, only this base station can choose a free channel from g_j until that station frees the group. Since the group is not freed until Ch_f has been chosen, only this base station can use Ch_j.

Theorem 2 *There is no (individual) starvation in the DDRA algorithm.*

Proof: By Lemma 1, if there is a free group, a requesting base station B_i eventually gets a free group. This process continues until either B_i obtains a free channel or exhausts the list of free groups. In either case, B_i exits section A of the DDRA-algorithm. Since no other section of the algorithm contains a loop, B_i will exit them as well. Hence, B_i will never starve.

3.4.1 Message complexity of the DDRA-algorithm

Let us assume the worst scenario, i.e., only one group g_j can have a free channel Ch_f, and that base station B_i visits all of the other groups before visiting g_j.

As illustrated in Section 3, the last iteration starting with step A1, the base station B_i will eventually visits the group g_j. Hence, there are N_g iterations where N_g is number of channel group. In order to get permission from its neighbors, B_i needs $2 * (N_n - 1)$ messages in each iteration i where N_n is the number of neighbors. Moreover, in the final iteration, to update information about the channel Ch_f, it needs to send $(N_n - 1)$ message to all of its neighbors. In the worst case, there are

$(N_g - 1)logN_n$ competitions [3]. Hence, the total number of messages is $2 * (N_n - 1)(N_g - 1)logN_n + (N_n - 1) = O(N_g * N_n logN_n)$.

3.5 Simulation Environment for DDRA Algorithm

In order to study the performance of the algorithm, a discrete-event model [52] is designed to simulate a wireless network of 57 cells. In the experiments, the number of channels are varied from 50 to 400. The DDRA's performance is evaluated under two environments: *uniform* and *non-uniform* call arrival distributions. In the first model, a uniform distribution is assumed, where each cell has the same channel demand, i.e., call arrivals to a mobile host (MH_i) are equal to λ, and the service time are equal to μ. In the second model, one third of the cells is used as heavily loaded cells. Lightly loaded cells have call arrival rates 1/5 that of the heavily loaded cells (*hot* cells). Several schemes have been investigated in order to select the heavily loaded cells. This dissertation used the following two strategies:

- non-uniform even distribution, in which heavily loaded cells are evenly distributed over the region.
- non-uniform random distribution, in which any cell can be selected as a heavily loaded cell.

In all cases, the messages that are sent by the Base Stations contain the following information: *Packet type, Source BS, Destination BS, Competition_num, Channel_num*. We use five types of packets [4]. As illustrated in Table 3.1, in the simulation environments, assume that the average one way communication delay between two BSs is 0.03 mSec, which covers the transmission and propagation delays, and that it takes 2 mSec to process the packet. Thus, the mean length of communication time to send a packet is 4.03 mSec. The call durations are modeled by an exponential

[3] Note that the competition of getting a free group creates a binary tree structure with N_n leaves and a height of $logN_n$ levels.

[4] Hence, 3 bits are enough to identify each packet. Thus, the size of the message is at least $3 + 2log(N_n) + log(N_n) + log(N_c)$ bits where N_n is the number of neighboring cells and N_c is the number of channels in the system. Hence, We need at least 32 bits if 400 Channels are used.

Number of cells	57
Number of channels	400, 100, 50
Mean service time /session	180s
Arrival rate in normal cell	λ
Arrival rate into a hot cell	5λ
Mean delay to processing a msg.	2.0 mSec
Mean msg. delay between BSs	0.03 mSec

Table 3.1: Simulation Parameters

process with a mean μ, and the call arrivals by a normal process with an arrival rate λ. The call arrival rates for *h*eavily loaded cells keep equal to 5λ. The mean service time per communication session is set to 180 Sec. In order to avoid the congestion problem, Little's law is used to compute the system load, i.e., the channel request load from MHs. According to Little's law, the system load that used to compare the performance of the channel systems can be computed.

We assume the communication line has 1 Mbits/Sec bandwidth between two BSs. Experimental evaluation of channel assignment strategies requires not only an implementation but also the examination of a large number of test cases. A large number of parameters also complicates the performance evaluation. Consequently, a complete, factorial testing of all combinations is clearly prohibitive.

In the course of the experiments, the system load [45] is defined as the average number of call arrivals, and the performance of the algorithm is evaluated by using the following three metrics: (i)*Denial rate* which indicates the average fraction of unsuccessful requests; (ii)*Acquisition time* which measures the amount of time that a mobile host (i.e., user) waits until its requests can be granted. This metric takes into consideration several factors such as transmission and message processing delay, just to mention a few; and (iii) *message complexity* which measures the overhead of the algorithm in terms of the number of messages needed to satisfy the user's channel requests.

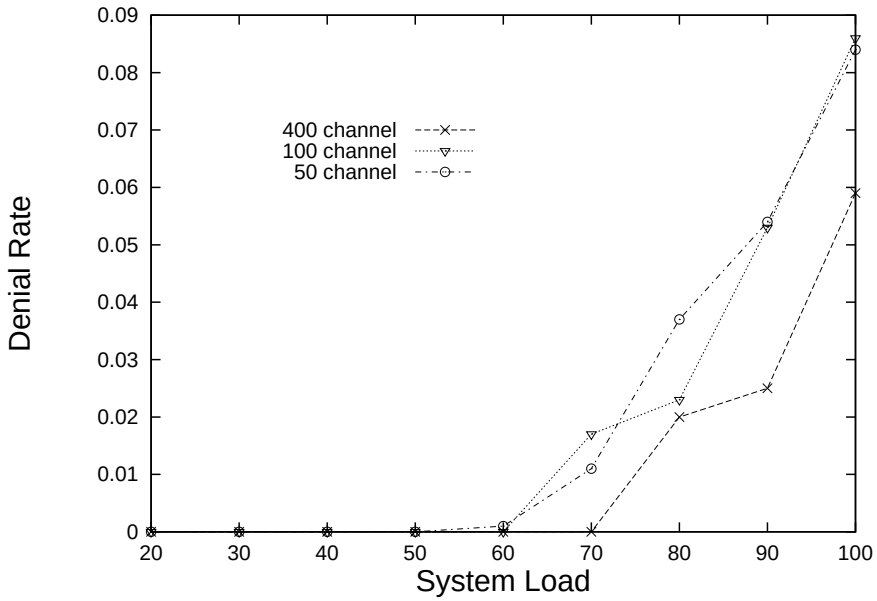

Figure 3.6: Denial Rate Vs System Load Using the Uniform Distribution Model

3.5.1 Experimental Results

In this section, the performance of the algorithm, and assess the denial rate and the acquisition time obtained by the algorithm are evaluated. Then, the message overhead implicit in the use of the DDRA channel allocation scheme is discussed. The experimental data that follows were obtained by averaging several runs with an interval of confidence between 90-95%. Recall that the denial rate represents the average fraction of unsuccessful requests and the acquisition time represents the response time to a free channel.

Let us now turn to the results. Figures 3.6 and 3.7 portray the denial rate obtained for the uniform and non-uniform traffic distributions, for three channel levels ranging from 50 to 400 channels. The algorithm exhibits a very low denial rate (i.e., less than 10%) when the system load is less than 100% for all traffic models and for all channel levels. This is largely due to DDRA's efficiency in granting channel requests as fast as possible. As expected, the results also indicate that increasing the number of channels will result in decreasing the denial rate.

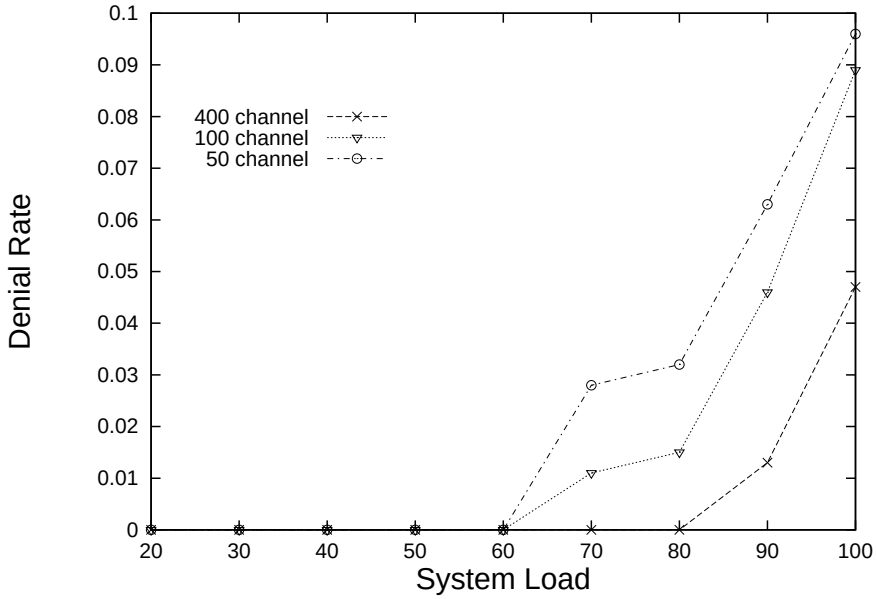

Figure 3.7: Denial Rate Vs System Load Using the Non-Uniform Even Distribution Model

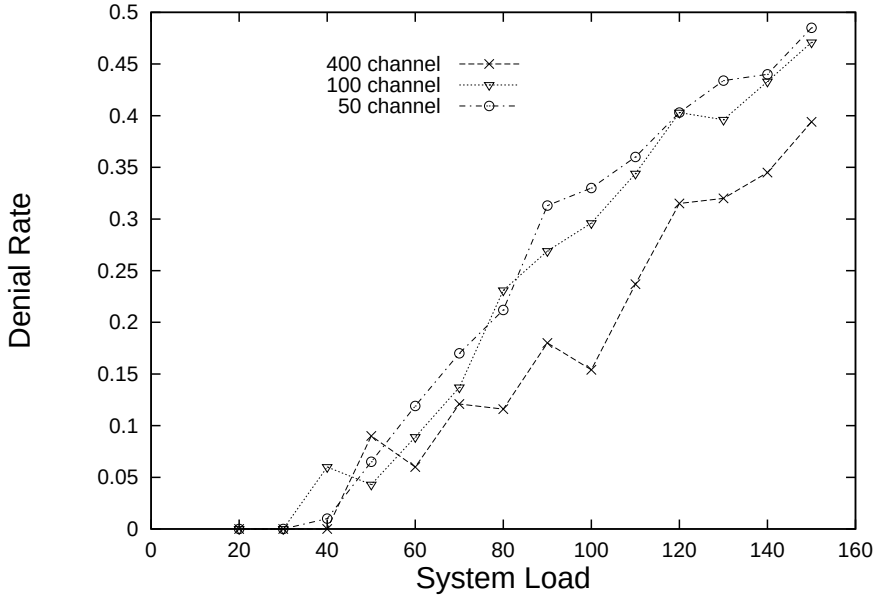

Figure 3.8: Denial Rate Vs System Load Using the Non-Uniform Random Distribution Model

In the case of a non-uniform random traffic model, the results are not as dramatic as in the uniform and non-uniform random models, (see Fig 3.8). This is mainly due to the hot cells' location and their impact on the denial rate. For instance, with the 400 channel system, less than 15% of the requests are denied when the system load is close to 100%. As expected, in all of the three cases, the denial rate increases by increasing the system load in all cases, and that the denial rate decreases as the number of channels is increased within the system.

Next, the average response time per channel request is measured. The results in the form of graph of the average acquisition time per channel of the model as a function of the system load employed to run the simulation models is presented. As we can see from the curves in Figures 3.9, 3.10, and 3.11, the response time to grant a channel to each request is at least 4.0 $mSec$, which represents the minimum time for a channel acquisition. This is mainly due to the fact that no cell holds any channel in the DDRA algorithm, i.e., in any cell, a BS can use any available channel in the system. The results indicate that a significant response time can be obtained using a system with 400 channels. These results mirror those obtained for the denial rate.

Figures 3.12, 3.13 and 3.14 portray the message overhead of the algorithm in allocating the available channels as the system load is increased. The messages complexity increases as the channel request load is increased, and it decreases as the number of channels is increased. We believe that this is due to the efficiency of the allocation scheme, in which most of the call requests are satisfied under a low channel request load. The average number of messages required to allocate channels using the algorithm about 5, 6 and 10, respectively, for uniform, non-uniform even, and non-uniform random traffic distribution models when the channel request load is close to 100. These results are very encouraging when DDRA to previous algorithms are compared, as the results of this shall be seen later.

Note that size of the message is bounded within the sender's neighbors. Hence, the number of message is independent of the size of the cells, and thereby of the number of channels.

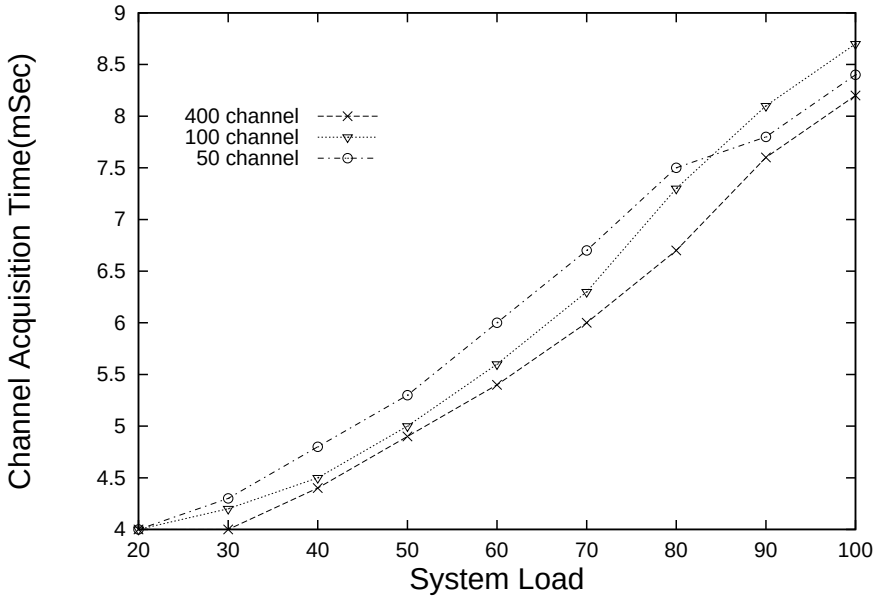

Figure 3.9: Acquisition Time Vs System Load Using the Uniform Even Distribution Model

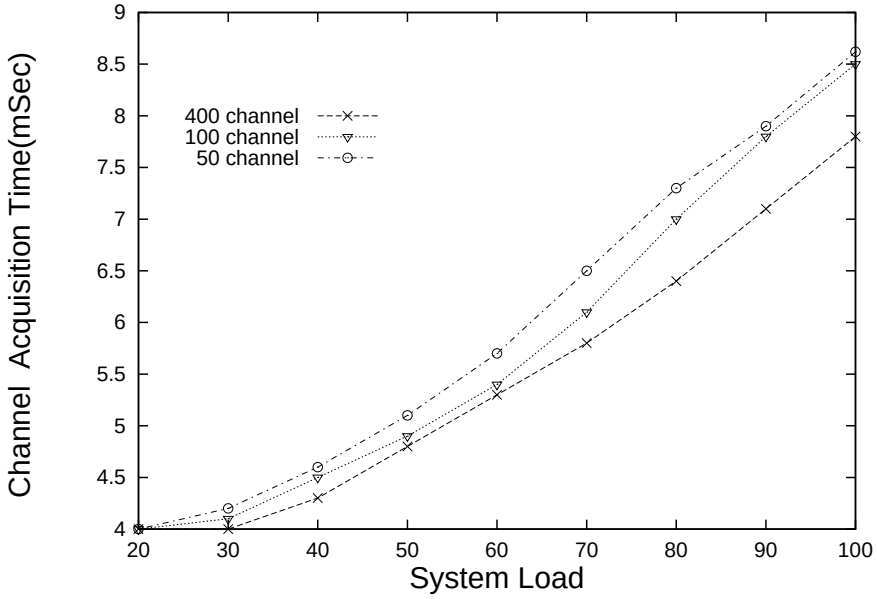

Figure 3.10: Acquisition Time Vs System Load Using the Non-Uniform Even Distribution Model

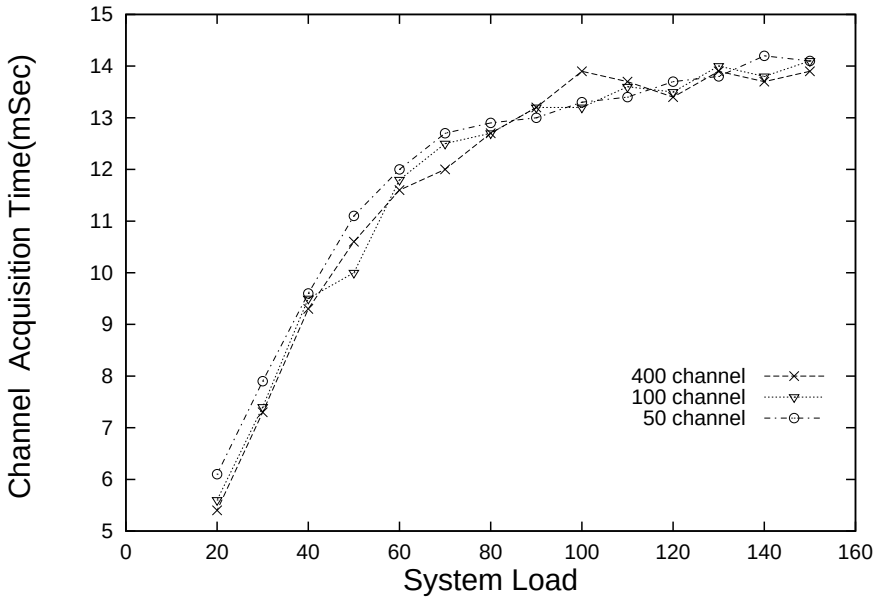

Figure 3.11: Acquisition Time Vs System Load Using the Non-uniform Random Distribution Model

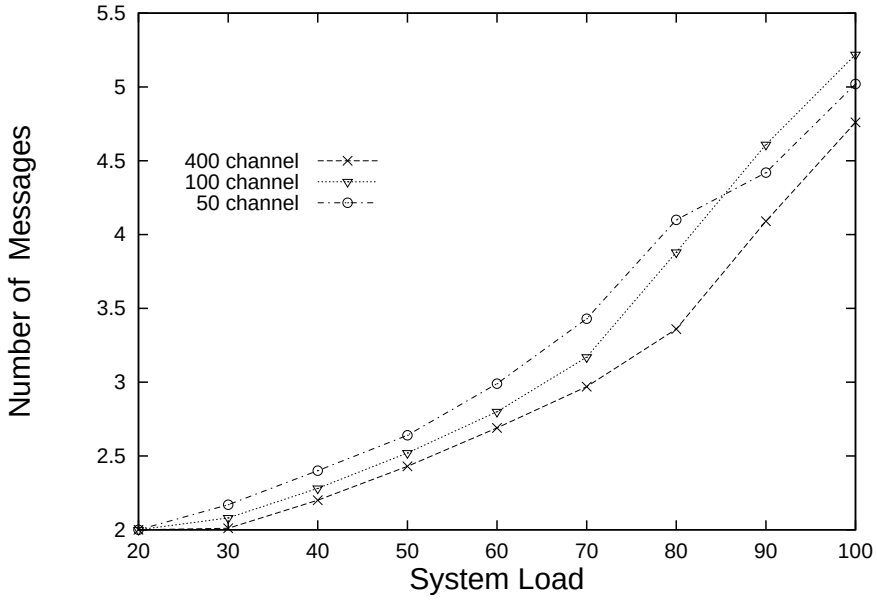

Figure 3.12: Message Complexity Vs System Load Using the Uniform Distribution Model

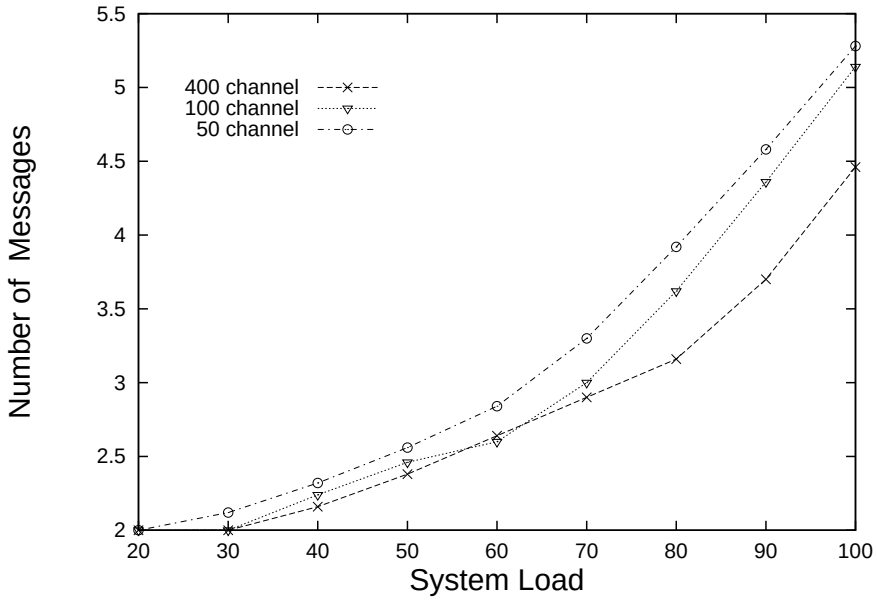

Figure 3.13: Message Complexity Vs System Load Using the Non-Uniform Even Distribution Model

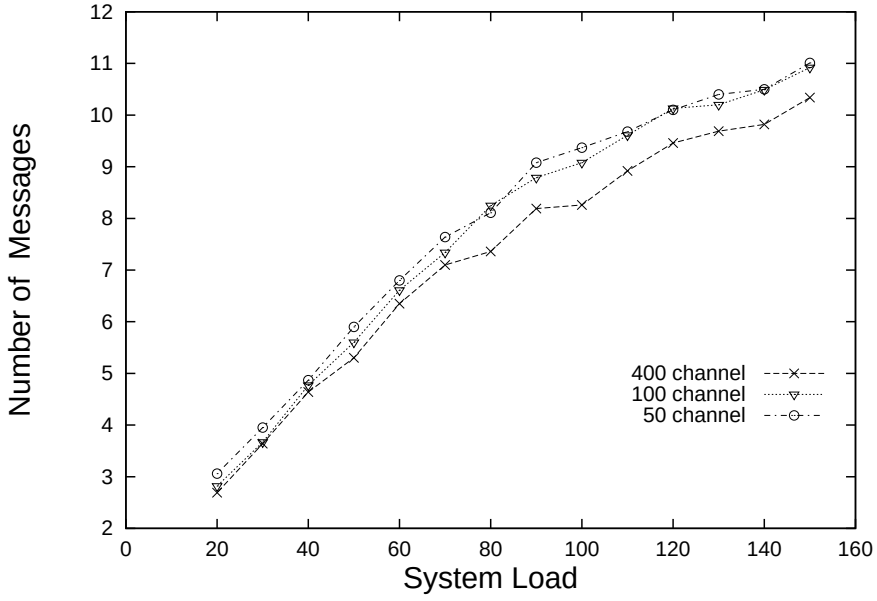

Figure 3.14: Message Complexity Vs System Load Using the Non-Uniform Random Distribution Model

3.5.2 Previous and Related Work- A Comparison for Channel Assignment Algorithms

In this section, the scheme with previous channel allocation methods (see Table 3.2) are compared. In [40], the channel assignment problem was analyzed as a generalized list coloring problem. The authors adopted a sequential solution to the problem [3], then they generalized it using a distributed algorithm using the double doorway and the synchronization mechanisms as suggested in [13]. The response time and the failure locality were the major performance measures used to evaluate their algorithms. In [14], the authors presented a channel allocation algorithm based upon the resource allocation problem in distributed systems. They used existing paradigms of mutual exclusion and dining philosophers. In order to evaluate the performance of their scheme, they used the response time and the successful channel assignment ratio. The results obtained were quite encouraging. In [45], the authors used timestamps to control the order in which the requests are processed. A base station that wants to get a free channel can use the channel, only after its neighbors have approved it. In their scheme, each neighbor knows the order of the packets it receives. Based upon this order, the base station issues its approval to its requesters.

In Table 3.2, the message complexity and the size of the messages involved in the above algorithms with the DDRA algorithm are compared. We have used N_g to denote the number of groups, and N_c to denote the number of channels in a system [5]. According to the results, the DDRA algorithm has a better message complexity.

3.5.3 Experimental Results for DDRA Algorithm

In this section, the simulation experiments is reported. the algorithm with the optimistic algorithm (Opt. Algo.) [45] and the pessimistic algorithm (Pes. Algo.) [13] are compared. These algorithms are chosen because (*i*) they are both distributed algorithms; (*ii*) they are known to have the best response time and denial rate; and (*iii*) they are based upon the same mutual exclusion paradigm. Two set of experiments

[5]Note that if N_g is used to represent the number of groups in the system then it will be $logN_n \leq N_g \leq N_n \ll N_c$ where N_n is the number of neighboring cells and N_c is the number of channel in the system.

Authors/Algorithms	Message Complexity
Choy [14]	$O(N_g^2 * N_n)$
Naveen [40] Algorithm 1	$O(N_g * N_c)$
Naveen [40] Algorithm 2	$O({N_c}^2)$
Prakash [45]	$O(N_c * N_g)$
DDRA Algorithm	$O(N_g * N_n log N_n)$

Table 3.2: Comparison with existing algorithms

are performed using the system load of 400 channels. The first assumes a uniform request arrival rate over the entire region. The second assumes a non-uniform even distribution as described earlier. The metrics used for comparing the performance of the algorithm to the optimistic and pessimistic ones are the same to the ones used previously; i.e., denial rate and acquisition channel time.

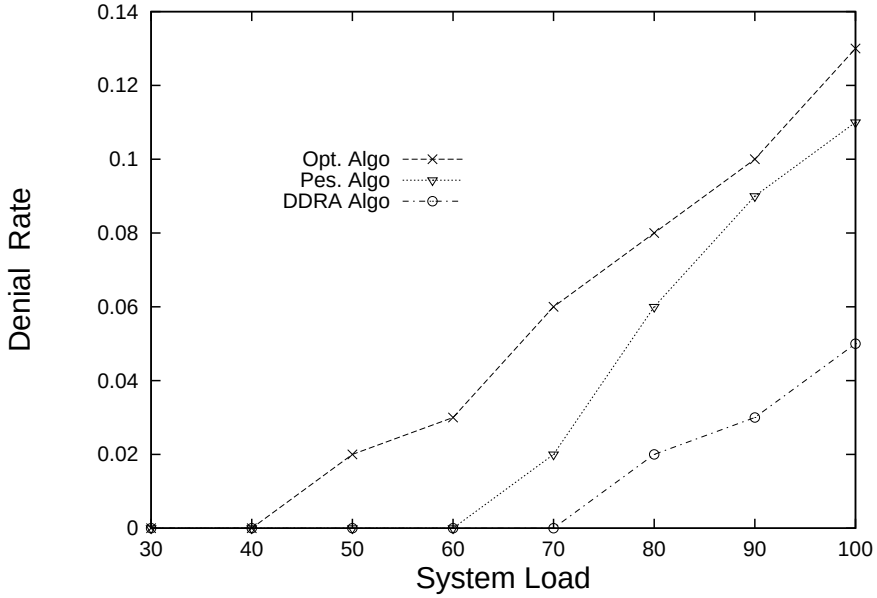

Figure 3.15: Comparison of Denial Rate Using the Uniform Distribution Model

Let us now turn to the results. The denial rate under the uniform distribution is shown in Figure 3.15. All three algorithms have almost the same denial rate under 70% system load, and their denial value is very low (close to zero). However, the results indicate that the DDRA algorithm has the lowest denial rate when compared

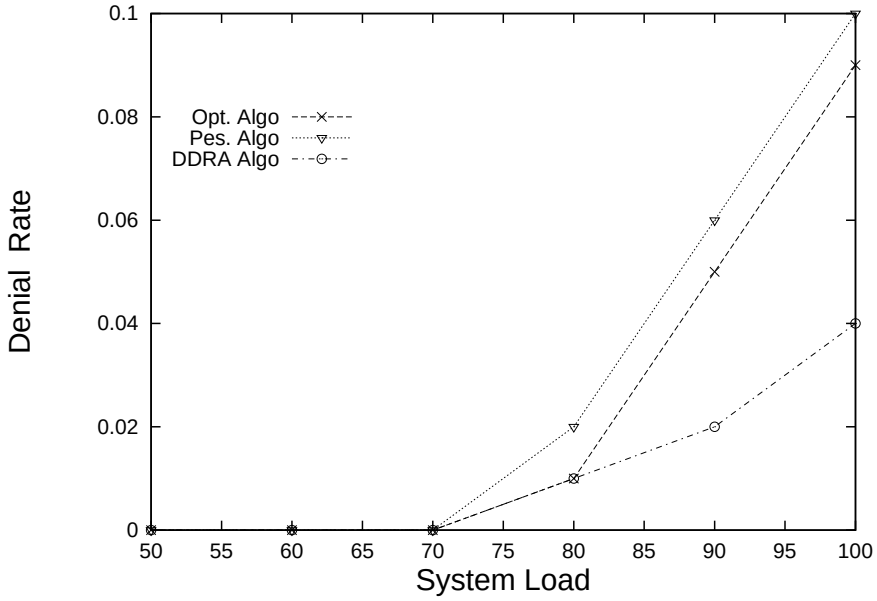

Figure 3.16: Comparison of Denial Rate Using the Non-Uniform Even Distribution Model

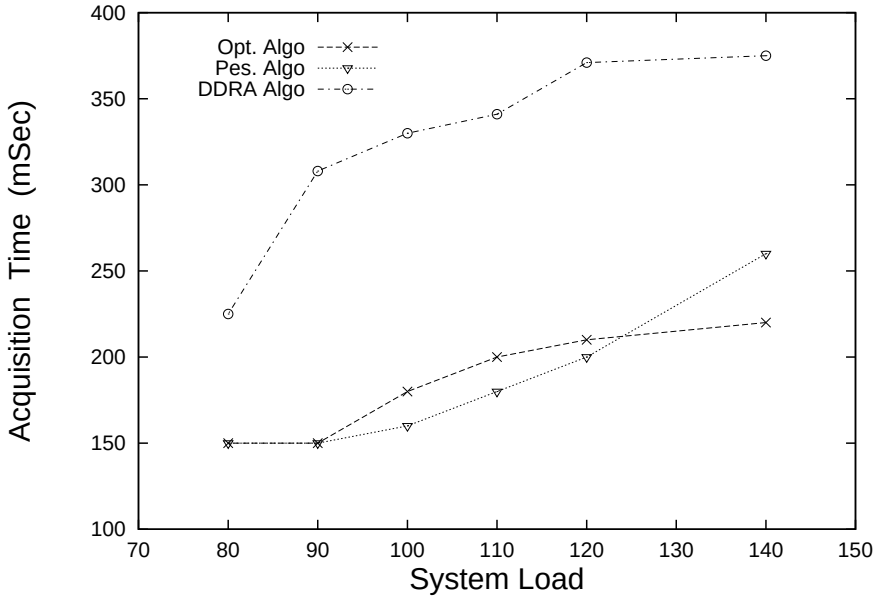

Figure 3.17: Comparison of Acquisition Time Using a Uniform Distribution Model

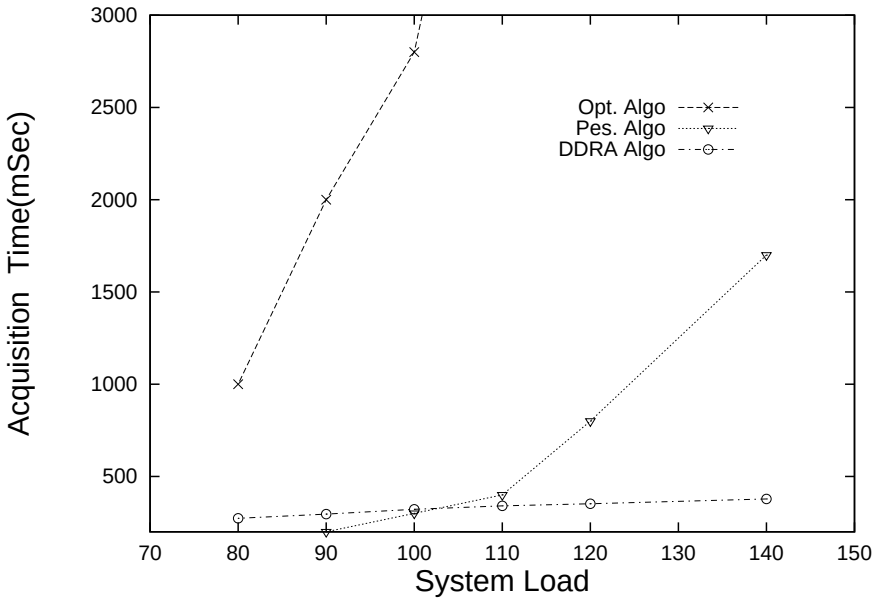

Figure 3.18: Comparison of Acquisition Time Using the Non-Uniform Even Distribution Model

to both the optimistic and pessimistic schemes with a system load of at least 80%. Furthermore, for all channel levels, an approximate 5% reduction in the denial rate using the algorithm over both optimistic and pessimistic algorithm when a system load of 80% or more is used. In Figure 3.16, the results obtained for the non-uniform even distribution are presented. The three algorithms have the same denial rate up to a system load of 100%, then the algorithm exhibits a better performance with the lowest denial rate. A 5% reduction in the denial rate can be obtained using the algorithm over both optimistic and pessimistic algorithm when a system load of more than 90% is used.

Figures 3.17 and 3.18 portray the results obtained for the acquisition time per channel requests under the two call arrival rates, i.e., uniform and non-uniform even distributions. Here again, the results mirror those obtained in the denial rate. The DDRA algorithm exhibits the best response time to grant and satisfy the channel requests with the non-uniform even distribution but it does not show good acquisition time with the uniform distribution. Furthermore, the acquisition channel time is always bounded under 400 mSec and under 500 mSec when respectively, the uniform

and the non-uniform even models are used. In either case, the DDRA algorithm has a bounded acquisition time while the other

algorithms have a non-bounded acquisition time (see Figure 3.18) because DDRA does not need to communicate to borrow a free channel from the neighboring base station as opposed to previous algorithm. If a group of channels is hosted to at a given base station, then the base station will borrow channels from its neighbors after a certain level. This is a very encouraging and promising results.

3.6 Conclusion

In this Chapter, the problem of dynamic channel assignment for wireless communication systems making use of the mutual exclusion model has been addressed. An efficient distributed algorithm for dynamic channel allocation is described. Its implementation is discussed and the results obtained to evaluate its performance using several channel systems and different types of call arrival patterns are presented. The algorithm requires less communication overhead than previous schemes, $O(N_g * N_n log N_n)$ messages Versus $O(N_g^2 * N_n)$ for Choy's algorithm, where N_g is the number of channel groups, and N_c is the number of channels in a system. We have presented an efficient distributed algorithm for dynamic channel allocation. We have discussed its implementation, and presented the results obtained to evaluate its performance using several channel systems and different types of call arrival patterns.

The results indicate that a 400 channel system produces the best results in all call arrival patterns (i.e., uniform, non-uniform even, and non-uniform random). A low denial rate, and low message complexity have been obtained in most of the experiments. Extensive simulation experiments were conducted to compare the performance of the algorithm with previous wireless channel allocation algorithms. The experimental results indicate that a significant reduction of the denial rate were obtained using the algorithm when compared to the use of optimistic/pessimistic schemes. For instance, a 5% denial rate improvement over previous algorithms where observed when a system load of 100 and a non-uniform distribution model are used. The results obtained also indicate that the acquisition time using the DDRA algorithm and

a non-uniform model with 100 system load, is at about 35-45% faster than previous algorithms. Last, but not least, the DDRA algorithm exhibits an upper bound of 400 mSec when a non-uniform model is used, where previous algorithms exceeded this when the system load is increased.

CHAPTER 4

MULTIMEDIA SYNCHRONIZATION SCHEME

Recent advances in wireless communication, multimedia and mobile computing technologies make feasible a new paradigm called mobile distributed multimedia systems (MDMS) that may simultaneously manipulate diverse media objects including text, images, audio, and video. Some of these objects are time dependent on each other when playing, but some of them are not. According to the time-dependency, these media objects can be divided into two groups: discrete objects that include text and images [41]; and continuous objects that include audio and videos [37].

The basic abstractions for time constrained media elements are timed streams of media components. The term "media component" means video frames, and audio samples. These components should be kept using a temporal order when playing them. The process of maintaining order of media components is called multimedia synchronization [8]. There are two kinds of timing aspects for time constrained elements: (1)intra-media continuity is subject to a real time constraint in handling media elements; (2) inter-media synchronization is subject to temporal correlation during a playback of media elements [44]. The synchronization problem under wireless/wired networks is more complicated than the problem for wired networks because wireless networks must use a base station to deliver packets, and have much less resources [22, 27]. Furthermore mobile/wireless communication system is a wireless network that provides low-power, and high-quality access to *mobile users* or *mobile hosts* (MH).

4.1 System Model

In this section, the system model is described and introduce the data model used here. The system consists of N scalable server nodes and M wireless clients (mobile hosts). Unlike in fixed networks, most of the time there is only one route between a base stations and a mobile host. A base station can directly communicate with

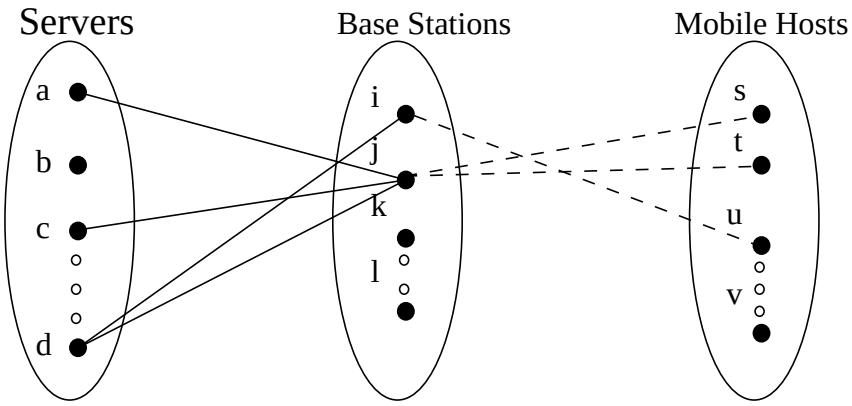

Figure 4.1: Distributed Multimedia Networks with Wireless Clients

many mobile hosts. Generally, as shown in Figure 4.1, the system contains a variety of servers, depending on the needs of the mobile hosts.

In the model, all mobile hosts get services from the same set of servers. Their communication is through base stations, where many servers connect to each base station, and each base station connects to many mobile hosts. Because of this, the base stations are assigned to represent mobile hosts within their cells.

A video consists of many video frames that can be divided into N equal-size parts, called *subframes*. *Subframes* are equally located at N different server nodes using a technique called subframe stripping [34]. During playback, wireless clients continually receive subframes from a server. Servers have admission control capability and also can change the data transfer rate at the mobile host's request.

The synchronization problem for multimedia objects is solved by considering the following three cases: (*i*)no jitter and constant network delay case; (*ii*)inter and intra-stream synchronization problems; and (*iii*) resynchronization. In the first case, the work of Biersack and Geyer [7] is extended at the dissertation and present a framework for wireless multimedia hosts. In the second case, Quality of Service (QoS) parameters is considered. In the last case, exponential smoothing forecasting for resynchronization purposes is used.

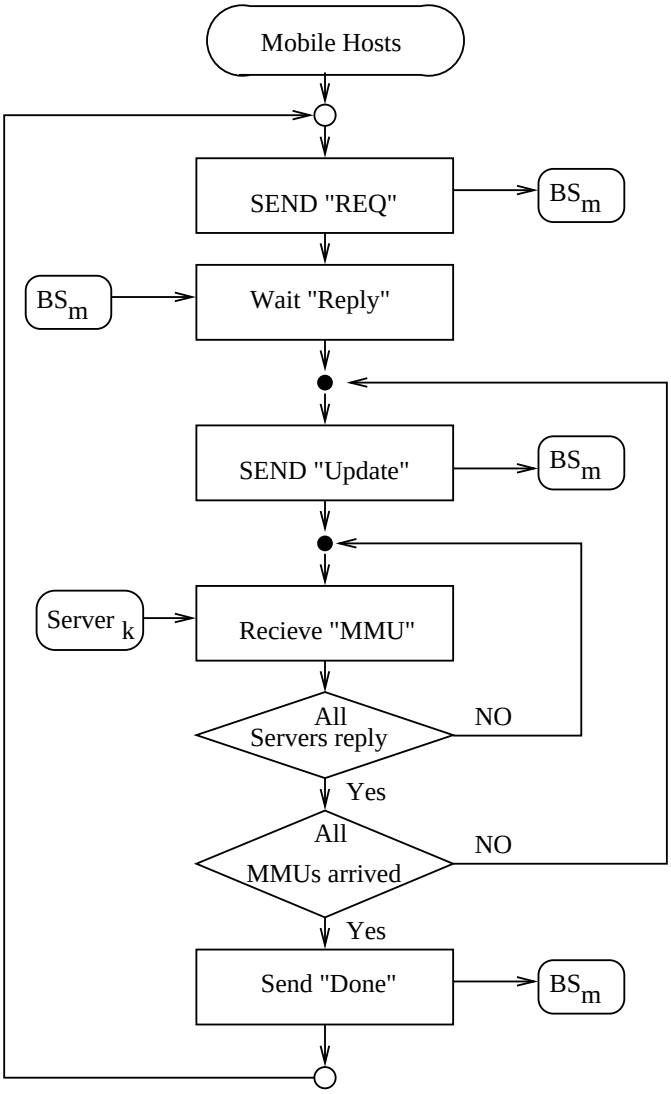

Figure 4.2: Basic Protocol for a Mobile Hosts

In this section, the mobile synchronization algorithm MoSync is described. Unlike other types of distributed schedulers, the MoSync algorithm uses three types of nodes: server, Quasi-receivers, and receivers. The model consists of K servers, N base stations, and L mobile clients, where $L,\ K,\ N > 0$. The K servers have type *server*, the M base stations type *Quasi-receiver*, and the L mobile clients type *receiver*. The

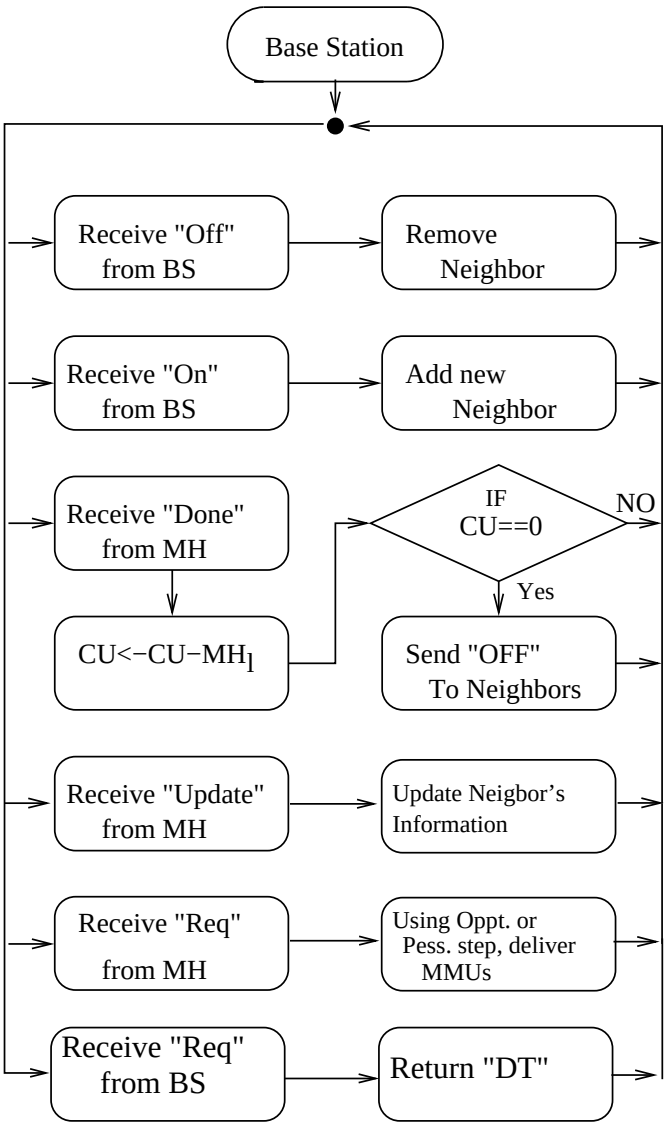

Figure 4.3: Basic Protocol for a Base Station

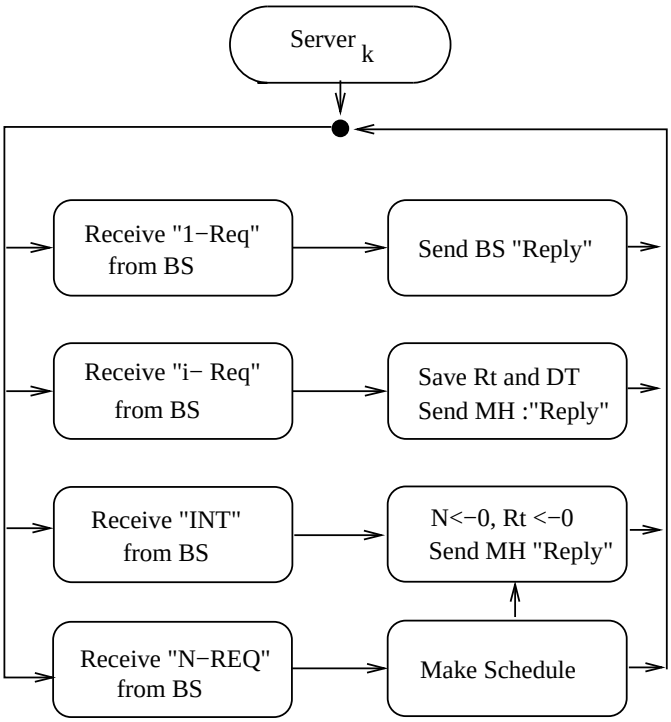

Figure 4.4: Basic Protocol for a Server

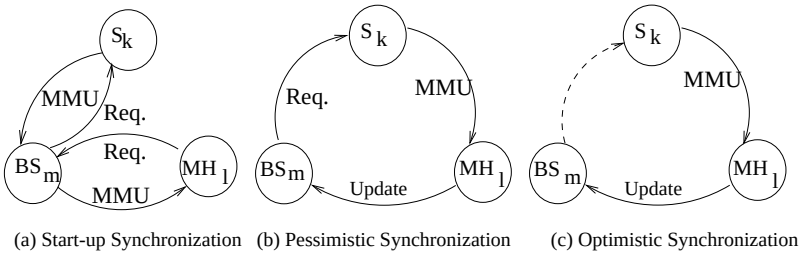

Figure 4.5: Simple Protocol

mobile hosts (Wireless Multimedia Clients) have the responsibility to synchronize the multimedia objects received from multimedia servers. Mobile hosts report the amount of buffer consumed to the base station that controls the area where they are located and also let the base station know the differences among multimedia objects' arrival times. After a base station (BS) has information about the arrival time of each multimedia object, it calculates the synchronized time for the next packets. A scheduler in the server manages the transfer of the subframes as part of the frames to the wireless clients on time.

MoSync has three parts, one for each type of host. As in Figure 4.2, MH_l requests multimedia service through BS_m to the server S_k. The base station has three roles: a *messenger* role, a *filter* role, and a *Quasi-receiver* role. The Base Station, acting as a messenger, passes multimedia packets to mobile clients. As a filter, it sends a request for packets to servers, and as a Quasi-receiver, it receives only the first packets from the servers. After that, the multimedia objects are sent directly from the senders to the mobile hosts. It also calculates the synchronization point for each server, *i.e.*, $T_{max} - T_i$, where T_{max} is the largest observed delay round trip time (RTT) from all servers and T_i is the observed RTT from the server S_i. When a base station requests the first group of multimedia objects, it sends the synchronization point information to all servers.

As in Figure 4.3, the base station receives messages from two sources, neighboring BSs and mobile hosts. The messages from mobile hosts consist of multimedia "requesting" message, information "update" messages and "Done" messages. The messages from neighboring BSs consists of "On" and "Off" messages. On receiving

the "requesting" message from a mobile client, if a base station's own current synchronization point is less than minimum delay time δ [1], it requests multimedia objects to servers without a set-up synchronization step. If it is bigger than δ, then the base station requests dummy packets [2] to make a fresh synchronization point, and then requests the multimedia objects as shown in Figure 4.2, 4.3, 4.4. On receiving the "update" message from a mobile client as shown in Figure 4.3, a base station gets message including the difference between the arrival time and the expected arrival time.

After the base station gets the message, it subtracts the last time difference from the new time difference. If the result is smaller than the minimum delay, then it increases the credit for the server that sent the message. If not, the credit will not be changed. Note that the credit reflects the status of the network; *i.e.*, delivery multimedia units. Accordingly, the base station will send a "Request" message or an "N-Request" message. The "Request" message request only a multimedia unit but "N-Request" does multiple multimedia units. If the N-Request is running already, then "N-Request" can be requested from the server only after either N multimedia units have arrived or the time difference is bigger then δ. On receiving the "Done" message from a mobile host, a base station will terminate the multimedia service. On receiving the "On" message from a neighboring BS, the base station will know that the set-up delay time is available on the neighboring BS. If the base station receives "Off" message, then it will know that the set-up delay time is not available on the neighboring BS.

As in Figure 4.3, servers will receive four types of messages: (*i*) a message to set up initial synchronization; (*ii*) a message to request one single multimedia object; (*iii*) a message to request N multiple multimedia objects; (*iv*) a message to interrupt delivery of N multiple multimedia objects. When a server receives an interrupt (INT) message from a base station, it terminates the delivery of the N multiple multimedia objects and it also eliminates all future deliveries scheduled for the base station. Then, with a new delay time, it sends out a multimedia object to the base station. When a

[1] Minimum delay time is a constant value chosen as a parameter of the algorithm.

[2] The packet contains only size of the packet, source and destination

server get an "N-Request" message from a base station, the server makes the future delivery schedule for the Mobile Hosts and sends the first multimedia object with it. Upon receipt of the first delivery, the server delivers multimedia objects every RT_i time.

When MH receives first multimedia object, it calculates the latest arrival time and the differences between every multimedia object arrival time and the buffer usage.

$$buffer\ usage\ =\ (NumMMUs) \bullet (PoutTime)\ -\ CurTime\ +\ ReqTime;$$

where *NumMMUs* is number of multimedia units, *PoutTime* is the play-out time for a MMU, *CurTime* is the current time at MH, and, and *ReqTime* is the time when the MH request MMUs.

4.2.1 Formal Description of Algorithm

We now formally describe the algorithm for a base station, BS_m, a mobile host, MH_l, and a multimedia server,S_k by means of pseudo-code. We provide a simple example to illustrate the features of the scheme. In the sequel, the notations are used as shown in Table 4.1 to present the scheme. The algorithm uses two types of message:

(1) Send: $\langle$ source, sink, referee, action; arguments$\rangle$;

(2) Receive: $\langle$ source, sink, referee, action; arguments$\rangle$;

where action $\leftarrow$ Request | Reply | Update | On | Off | Done; referee $\leftarrow$ Client | ϕ | Server; and arguments $\subset \{MS_k^i,\ DT_k^i,\ RT_k^i,\ TS_k^i,\ S_k,\ ST_k\}$

4.2.2 Pseudo-code

In this section, the algorithm for the mobile communication model described in section 4.2 is provided.

Server$_k$:

S1: **Upon Received** $\langle\ BS_m,\ S_k$,MH_l Request; $MS_k^0\ \rangle$

S2: Send $\langle\ S_k,\ BS_m$, Reply; $MS_k^0\ \rangle$

S3: **Upon Received** $\langle\ BS_m,\ S_k, MH_l,$ Request; $MS_k^i,\ DT_k^i\ \rangle$

S4: Save Rt_k^i and DT^i

S5: Send $\langle\ S_k,\ MH_l,$ Reply; $MS_k^i\ \rangle$

S6: **Upon Received** $\langle BS_m, S_k, MH_l,$N-Request;N,$MS_k^i, DT_k^i, RT^i\rangle$

S7: Make schedule for (N-1) requests// N requests

S8: Save Rt_k^i and DT^i

S9: Send $\langle\ S_k,\ MH_l,$ Reply; $MS_k^i\ \rangle$

S10: **Upon Received** $\langle BS_m, S_k, MH_l,$INT;$MS_k^i, DT_k^i\rangle$

S11: N $\leftarrow$ 0 // Terminate N-Request routine

S12: $RT^i \leftarrow 0$ // Reset round time

S13: Send $\langle\ S_k,\ MH_l,\ \phi,$ Reply; $MS_k^i\ \rangle$

MobileHost$_l$:

M1: Send $\langle\ MH_l,\ BS_m,\ S_k,$ Request; MM $\rangle$

M2: Receive $\langle\ BS_m,\ MH_l,\ S_k,$ Reply; $MS_k^0,\ \{\ DT_k^i \mid 1 \leq$ k $\leq$ K $\}\ \rangle$

M3: **Do**

M4: Send $\langle\ MH_l, BS_m,\ S_k,$ Update; $S_k,\ MS_k^i,\ \{\ DT_k^i \mid 1 \leq$ k $\leq$ K $\}\ \rangle$

M5: **Do**

M6: Receive $\langle\ MH_l,\ S_k,\ \phi,$ Reply;$MS_k^i\ \rangle$

M7: $TS_k^i \leftarrow$ Current Time

M8: $DT_k^i \leftarrow MP^i$ - TS_k^i // Difference from middle point

M9: **While** (All Reply)

M10: **While** (All MM)

M11: Send $\langle\ MH_l, BS_m,\ \phi,$ Done;$\rangle$

BaseStation$_m$:

B1: **Upon Received** $\langle\ MH_l,\ BS_m,\ \phi,$ Request; MM $\rangle$

B2: Set Lock On // Wait until Lock off : FCFS

B3: **If** (Time limitation over) // Is it good enough to

B4: **If** (Number of(NE_ON)) // Search for a neighbor from local data

B5: Send $\langle$ BS_m, BS_n, ϕ, Request; { DT_k^i | 1 $\leq$ k $\leq$ K } $\rangle$

B6: Receive $\langle$ BS_n, BS_m, ϕ, Reply; { DT_k^i | 1 $\leq$ k $\leq$ K } $\rangle$

B7: **Else**

B8: **For all** $S_r \in S_{1..K}$**do** Send $\langle$ BS_m, S_r, MH_l, Request; MS_k^0 $\rangle$

B9: **Do**

B10: Receive $\langle$ BS_m, S_k, MH_l, Reply; MS_k^0 $\rangle$

B11: $TS_k^0 \leftarrow$ Current Time

B12: **Wait** (Receive all dummy packets)

B13: $TS_{Max} \leftarrow$ Max $\{TS_k^0 \mid 1 \leq k \leq K\}$

B14: $DT_k^0 \leftarrow TS_{Max}$ - TS_k^0

B15: **End If**

B16: **End if**

B17: Set Lock off

B18: CU $\leftarrow$ CU $\cup$ MH_l // Add a new service requester

B19: **If** (Number of(CU) ==1) // Number of current mobile clients

B20: **For all** $BS_n \in NN_m$ **do**

B21: Send $\langle$ BS_m, BS_n, ϕ, On;$\rangle$ to BS_n

B22: **End If**

B23: Send $\langle$ BS_m, MH_l, ϕ, Reply; MS_k^0, { DT_k^i | 1 $\leq$ k $\leq$ K } $\rangle$

B24: **Upon Received**$\langle BS_m, MH_l, S_k$,Update;S_k, MS_k^i, { DT_k^i | 1 $\leq$ k $\leq$ K } $\rangle$

B25: **For** k=1 to k $\leq$ K

B26: **If** (ABS (DT_k^i - DT_k^{i-1}) $\leq \delta$)

B27: NReq_count++; // Increment number of packet

B28: **If** (N == 0)

B29: N $\leftarrow$ NReq_count; // Update repeat and reset count

B30: Send $\langle$ BS_m, S_k, MH_l, N-Request; N, MS_k^i, DT_k^i >

B31: **End If**

B32: N $\leftarrow$ N - 1; // Reduce current remained iteration

B33: **Else**

B34: **If** (NReq_count > 0) // to terminate Optimistic condition

B35: Send ⟨ BS_m, S_k, MH_l, INT; MS_k^i, DT_k^i ⟩

B36: **Else** // to start pessimistic communication

B37: Send ⟨ BS_m, S_k, MH_l, Request; MS_k^i, DT_k^i ⟩

B38: **End If**

B39: NReq_count ← 0; // Reset number of repeat

B40: **End If**

B41: **EndFor**

B42: **Upon Received** ⟨ BS_m, MH_l, ϕ, Done; ⟩

B43: CU ← CU - MH_l

B44: **If** (Number of(CU) ==0) //Number of current MH

B45: **For all** $BS_n \in NE_m$ **do** Send ⟨ BS_m, BS_n, Off;⟩ to BS_n

B46: **End If**

B47: **Upon Received** ⟨ BS_m, BS_n, ϕ, On; ⟩

B48: NE_ON ← NE_ON ∪ BS_n //Add a new working BS

B49: **Upon Received** ⟨ BS_m, BS_n, ϕ, Off; ⟩

B50: NE_ON ← NE_ON - BS_n //Withdraw from service

B51: **Upon Received** ⟨ BS_m, BS_n, ϕ, Request; { DT_k^i | 1 ≤ k ≤ K } ⟩

B52: Get the latest delay time, DT_k^i;

B53: Send ⟨ BS_m, BS_n, ϕ, Reply; { DT_k^i | 1 ≤ k ≤ K } ⟩

4.3 Proof of Correctness

In this section, proving scenarios divided into three cases: (*i*) No jitter and constant delay; (*ii*) jitter; and (*iii*) clock drift, changing network conditions, and severe dropout.

*Case*1 : **No jitter and constant network delay case**
To avoid buffering early-arrival multimedia sub-streams, the "start-up" synchronization shown in Figure 4.2 is used to evaluate round-trip delays for the different sub-streams.

Lemma 3 *If there are only constant delay and zero jitter on the network, Algorithm MoSync makes the first L dummy units received from the servers to be a synchronization group for delivery to the mobile clients.*

Proof: Base station, BS_m, interconnects mobile clients and distributed multimedia servers, S_k where 1 $\leq$ k$\leq$ K. MoSync uses BS_m as quasi-sink to the all servers. Initially, at time TS_l^0, MH_l requests multimedia units from the server S_k. After BS_m gets a "Request" message from MH_l, BS_m multicasts it to all S_k at time t_0. S_k replies with a dummy message to BS_m. BS_m stamps the time, TS_k^0 on all dummy messages from S_k. BS_m then chooses the server response with the latest arrival time, TS_{max}. BS_m forwards the dummy message to MH_l with the difference, DT_k^0, between TS_k^0 and TS_{max}, MH_l sends an "Update" message indicating the observed difference, DT_k^0. This message causes BS_m to send a "Request" message to server S_k for the next multimedia unit with delay DT_k^0. S_k then sends a multimedia unit to MH_l directly. When MH_l gets a multimedia packet and it updates the packet arrival time. and sends an "Update" message to BS_m with the difference between the new arrival time and expected arrival time. Using this value, BS_m initiates a second iteration either by sending a new delay to S_k or by remaining silent if the delay has not changed. Each time S_k replies to MH_l directly after adjusting its delay appropriately.

*Case*2 : **Inter and Intra-stream Synchronization case**
In this section, end-system jitter and network jitter is solved. We assume that the

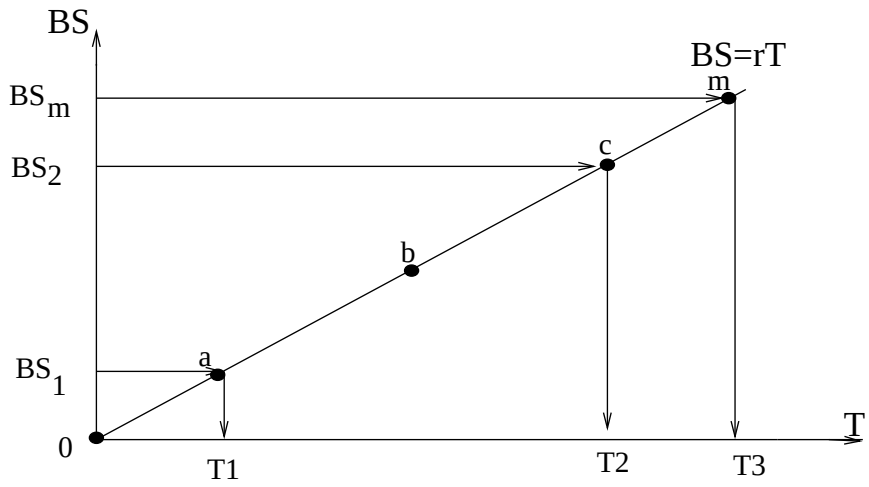

Figure 4.6: Smooth play-out range at $mobileclient_m$

jitter is bounded. Jitter can destroy temporal relationships within one sub-stream, and it can also change time gaps between arriving media units. Furthermore, the relationship between media units of a synchronization group can be changed. Therefore, inter-stream and intra-stream synchronization must be solved to deal with both end-system and network jitter problems.

Each mobile host MH_l has a message buffer of finite size b. Suppose that MH_l has a constant play-out rate of r multimedia units per second. If its servers are not able to keep the buffer sufficiently full, then MH_l will not be able to play out smoothly. Similarly, if the servers sent multimedia units too fast, the buffer will overflow.

Suppose that the buffer is partially filled. Let t be the point in time at which the buffer will be empty if no more multimedia units arrive; let T_1 be the time by which more multimedia units must arrive to guarantee smooth play out, given the bounded jitter; and let T_2 be the earliest time at which the buffer could overflow, given the jitter.

Suppose further that there are K servers S_k sending multimedia streams to MH_l, that each stream has length I, and that d_k^i is the delay of the i^{th} unit sent by S_k. Let $d_k^{max} = max\{d_k^i | 1 \leq i \leq I\}$ and $d_k^{min} = min\{d_k^i | 1 \leq i \leq I\}$. The jitter $\delta_k = d_k^{max} - d_k^{min}$.

Theorem 3 *Suppose that the servers S_k are serving the single mobile host MH_l and there is bounded jitter. The mobile host plays-out its multimedia units smoothly without buffer overflow whenever the servers S_k obey the following two constraints: (a) $t \geq T_1 \geq 2\, d_k^{max}$ and (b) $t \leq T_2 + 2\, d_k^{min}$*

Proof: MH_l plays-out its multimedia units with constant rate r as shown in Figure 4.6. In MoSync, the server S_k sends multimedia units to MH_l only as a result of an initial "Request" message issued from BS_m. Consider the worst case delivery time. There are two messages, a "Request" and a "Reply", taking $2\ d_k^{max}$ total time for delivery. Therefore, t cannot be allowed to fall below this figure, *i.e.* $t \geq T_1 \geq 2\ d_k^{max}$. In the best case where it takes $2\, d_k^{min}$ for both the "Request" and "Reply" messages to be sent, overflow may occur if MH_l does not play out its current units by time T_2 plus the minimum time for the next unit to arrive, i.e. $t \leq T_2 + 2\, d_k^{min}$.

Theorem 4 *Consider the case that delay is bounded between d_k^{max} and d_k^{min} for each server S_k. The buffer space at MH_l required to guarantee intra-stream synchronization is at most*
$\lceil 2\, \delta_k\, r \rceil$ media units.

Proof: Again, consider the worst case delivery time. Since it takes up to $2d_k^{max}$ time to receive the next multimedia unit, MH_l must be able to store at least $2d_k^{max} r$ units to avoid play-out delay. On the other hand, the best case delivery time is $2d_k^{min}$ so the buffer requirement is at most $2d_k^{min} r$ units. Hence for a single substream the buffer requirement is $\lceil 2\, \delta_k\, r \rceil$.

Theorem 5 *Consider multiple substreams in the case of bounded jitter between d_k^{max} and d_k^{min}. MH_l smoothly plays-out its multimedia units at a rate r whenever the following considerations hold:*
(a) Intra-stream synchronization is guaranteed,
(b) Inter-stream synchronization is guaranteed,
(c) Buffer space for N substream synchronization is $\lceil\ N2\delta^{max} r \rceil$
(d) The conditions in Theorem 1 hold.

Proof: Let d_k^i be delay of i^{th} element in the stream issued by S_k Let d_k^{max} = max $\{d_k^i \mid 1 \leq \text{i} \leq \text{I}\}$ and d_k^{min} = min $\{d_k^i \mid 1 \leq \text{i} \leq \text{I}\}$ where I is number of units in each stream. Let the jitter $\delta_k = d_k^{max}$ - d_k^{min}. δ^{max} = max $\{\delta_k \mid 1 \leq \text{k} \leq \text{N}\}$. Case (a)follows from theorem 4, and case (b) follows from lemma 3. For case (c), after MH_l has received the first unit from all N servers, it sends an "Update" message to BS_m and BS_m, in turn, sends "Request" messages to the servers. We will be able to guarantee interstream synchronization if there is sufficient buffer space. We can calculate this space using theorem 4:

*Case*3 : **Resynchronization**

Under the communication model suggested here, BS_m can control the number of units by sending messages only at selected times. To solve the average delay changes , clock drift, and server drop-outs, the following solution that uses exponential smoothing forecasting is suggested.

When MH_l gets multimedia units, it reports to BS_m the difference between the arrival time and the expected time of a unit. BS_m sends the difference to S_k, and S_k then sends out new multimedia units. This method has several drawbacks. Many returning packets contains very similar delay time information. Using simple delaying time to decide the next send time may not be good enough to make multimedia packets arrive within the required time. In addition, although the size of a feedback packet is small, many packets can cause congestion at the source, quasi-sink, and sink. These problems can not solved easily under the network algorithms considering source and destination only.

Using exponential smoothing forecasting from the protocol above, the returned value of $Delay_r$ is crucial to the next synchronization. Also, there is the possibility that this delay will change during the delivery process. Therefore an expected value for the next arrival time is used as

$$E_t = (1 - \alpha)\, N_t + \alpha\, OE_t$$

where E_t is expected arrival time of a packet from S_k to MH_l, N_t is new arrival time of the packets from S_k to MH_l, OE_t is old expected arrival time, α is smooth constant ($0 \leq \alpha \leq 1$)

4.4 Message Complexity for MoSync Algorithm

This seems to be the first scheme to consider a combination wireless/wired multimedia network. For completion, the MoSync algorithm with the source-sink model of Biersack and Geyer [7] are compared.

The comparison shows the algorithm to be efficient in message complexity because information about delay times is shared.

Theorem 6 *If L is the number of mobile hosts requesting multimedia service through base station BS_m, K is the number of servers processing those requests, and I is the number of multimedia units requested by each mobile host, then the message complexity of algorithm MoSync is $O(I(K+L))$.*

Proof: In the initial round each mobile host sends one request message to BS_m, which then sends a message to each of the K servers. The servers reply with K dummy messages, one per server. BS_m then sends a "Ready" message to each mobile host. The total messages for the first round is $2K + 2L$. No multimedia unit is sent to any mobile host in this round.

In subsequent rounds, there are at most L "Requesting" messages from the mobile hosts, K messages from BS_m to the servers, and L media units from the servers to the mobile hosts. Since there are I of these rounds, there are $I(2L + K)$ messages total in the I rounds. Hence the total number of messages sent in the worst case is $2K + 2L + I(2L + K) = O(I(K + L))$. We note, without proof, that if message delays are constant then the total number of messages sent will be only $2(I + 1)K + 2L$, since no message will be sent from BS_m to any server after the initial round. The message complexity of the sink-source algorithm [7] is $\Theta(IKL)$, which is worse than that of the algorithm.

4.5 Simulation Experiments

We have developed a discrete-event model to simulate a cellular wireless multimedia system on a wireless network and wired network. We assume that wired networks are used for communication between BSs and multimedia servers, and that cellular wireless networks are used for communicating between multimedia MHs and BSs.

4.5.1 Simulation Environment

In the model, there are 300 channels available for 60 cells, with system load equally distributed over all cells. Each multimedia hosts has buffer space at least three times as large as the largest MMU delivered from any server and at most six times as large.

Experimental evaluation of the synchronization scheme requires the examination of several test cases: many types of jitter and resynchronization. We evaluate MoSync's performance in two environments: *uniform* MMU arrival distributions and *non-uniform* MMU arrival distributions. As shown in Table 4.2, in the simulation experiments, both uniform and non-uniform distributions are used.

In the uniform case, the performance of the scheme assuming uniform MMU delay under jitter is evaluated. All cells have the same multimedia unit demand, and the requests from the mobile host (MH_i) have inter-arrival time λ and mean service time μ. The algorithm is simulated over four uniform MMU delay ranges: (A) 0 - 20 mSec, (B) 20 - 40 mSec, (C) 40 - 60 mSec, (D) 60 - 80 mSec. MMU delay times for all multimedia requests are evenly distributed between the minimum and maximum delay times, with the minimum delay time for delivery assumed to be 50 mSec in each case. The different jitter ranges have different effects. For example, jitter ranges A and B will cause overflow after some point. Unlike ranges A and B, range D will cause underflow.

In the second, non-uniform case, an exponential distribution is assumed. For this model, the algorithm is simulated over three MMU delay ranges: (A) 0-200 mSec, (B) 0-400 mSec, (C) 0-600 mSec. The ranges are exponentially distributed with mean values of 20, 40 and 60 mSec, respectively. An upper bound is set for each non-uniform distribution to avoid failure caused by very large delays in delivery from

MMUs to MHs. The mean communication session is set to 20 MMUs, and mean buffer size of a MH is set to 3 times the size of a MMU. These ranges will also exhibit different behaviors. For example, no case causes overflow, but cases B and C will cause underflow.

The performance of the algorithm is evaluated using the following metrics:

- *message complexity*, which measures the overhead of the algorithm in terms of the number of messages needed to satisfy the user's multimedia requests.
- *buffer usage*, which measures the synchronization behavior for each MH;
- *Overflow rate*, which measures the average number of multimedia units (MMU) that overflow a mobile client's buffer.
- *Underflow*, which measures the average deficiency in the client's buffer of the number of MMUs currently needed for smooth play-out.

The performance of the algorithm with variable-sized buffer space for each mobile host, and a variable number of requests from the MHs is evaluated. The following experimental data was obtained by averaging several runs. Recall from Section 3 that MoSync's message complexity is much better than that of the sink-source model, and the buffer space always holds the frames so that they can be used to play-out smoothly as long as MH requests multimedia frames.

4.5.2 Experimental Results

In this section, a set of experiments is divided into three parts. In the first part, the start-up system behaviors has been implemented. In the second part, MoSync with only inter-synchronization without any assumption of a peer-to-peer protocol is simulated. In the third part, inter-intra synchronization plus a control feedback mechanism that controls buffer availability and reduces the number of messages between source and Quasi-sink is implemented. In both parts, MoSync using the above metrics is evaluated.

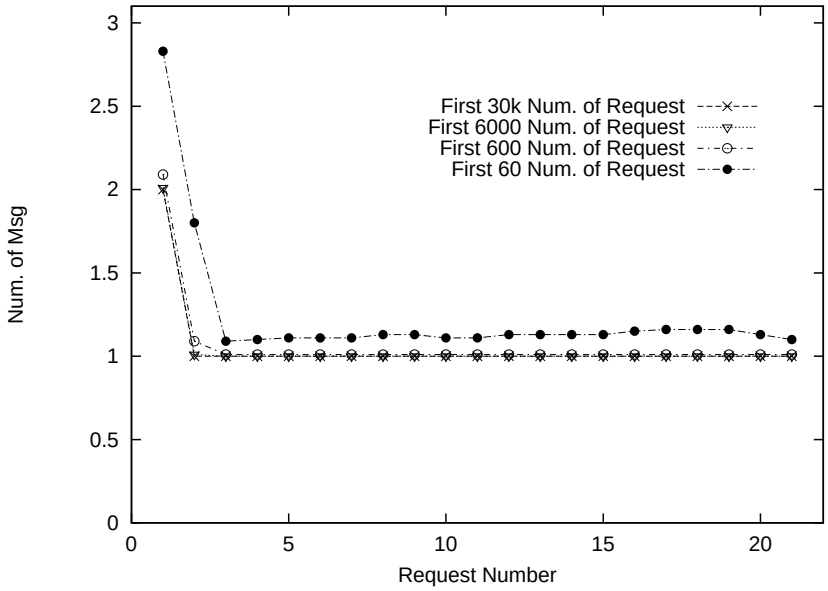

Figure 4.7: Number of messages Vs MMU number

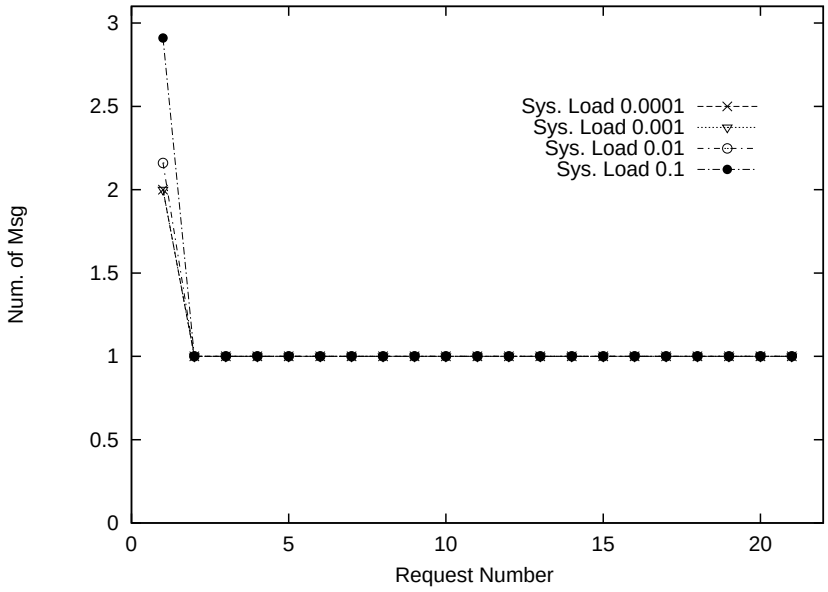

Figure 4.8: Number of messages Vs MMU number

• **Part A**

In the course of the simulation, the performance of the algorithm with a fixed size of buffer space and an upper bound, and assess the buffer space for each MH and the messages overhead due to the scheme to guarantee the synchronization of the MMUs among the mobile hosts has been evaluated. The experimental data that follows were obtained by averaging several runs with an interval of confidence between 90-95%. Recall that the systems can handle the situation where the buffer space should be able to hold the frames at any time so that they can be used, for a play-out for example, as long as an MH requests the multimedia frames.

Let us now turn to the results.

Figure 4.7 portrays the message overhead (request/reply messages) of the algorithm when MHs request (and reply) multimedia packets through the base stations. The message overhead decreases as the number of MMU request is increased, as well as the MMU request number. In the course of these experiments, we have noticed that in the first stage, (i.e., initial request), MH does not get any information from its base station. Indeed, at that time, the base station does not have any information such as servers' delay time, which is basically the main reason for delaying the servers. As we can see, the first 60 number of Requests require only a few more messages when compared to the other set of experiments. The results also indicate that an average message overhead of 2, when less than 6000 MMU requests are used. (Recall that the Requests are distributed evenly over all cells.)

After this initial stage, the results indicates that the message overhead is quite low but without jitter better results are produced. Note that the results obtained by using a pessimistic synchronization. Figure 4.8 portrays the results obtained as the system load is varied. As we can see the results obtained mirrors these obtained in Figure 4.7.

In Figure 4.9, the performance of the scheme is studied with respect to the buffer space usage, based upon the multimedia units requested. As we can see the buffer space usage, increases as the system load is increased as well as the request number up to 23, then it falls down . Without any jitter, a simple mechanism is used to avoid

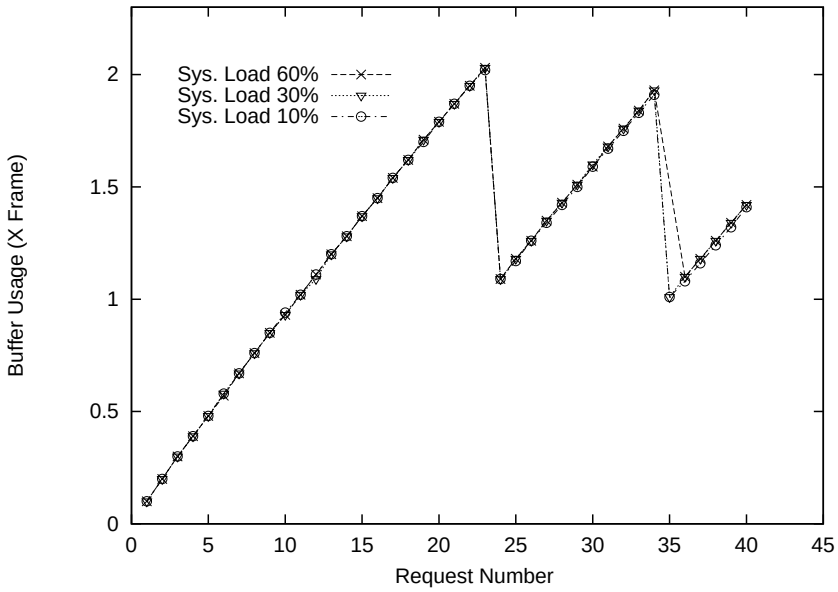

Figure 4.9: Buffer Space Vs MMU number

buffer overflow at MHs by adding a delay time under a pessimistic synchronization paradigm. As we can see, the buffer space usage is about 1.5-2 times of the frame size at the most. Recall that, so far in all of the simulation experiments, no jitter is assumed. The results indicate that promising results can be obtained in a no jitter case when compared to the the sink-source model especially at the start-up synchronization stage.

In the next set of simulation experiments, the performance of the scheme is evaluated using the jitter with a uniform delay scenario and without a resynchronization scheme. We used the following four uniform MMU delay ranges: [5,15] mSec for range A, [15,45] mSec for range B, [30,60] mSec for range C, and [45,75] mSec for range D. We assume that the distribution of MMU arrivals as exponentially distributed, and hence the base stations calculate the mean value with a delay time. BS inform the size of buffer space required to avoid buffer overflow and underflow at each MH and also the confidence of distribution can be set, i.e, the confidence is 95% for the results. Each MH adjust the size of buffer according to the information from its

BS.

The results obtained are summarized in Figures 4.10 and 4.11 where a uniform distribution is assumed. Figure 4.10 displays the results obtained for the buffer underflow rate as the number of requests are varied, and each request will bring four frames from servers. As we can observe, the results indicate that there is buffer space underflow in all of the level of ranges, with the exception of range D. We believe that is mainly due to the delay time which is much longer in the range D when compared to other level of ranges. Furthermore, not that MH overcomes the underflow only after it gets several buffer space that are underflow. These results indicate that the synchronization scheme must have a resynchronization function in order to reduce the rate of buffer space underflow.

Figure 4.11 displays the results obtained for the buffer space overflow rate as the number of requests are varied. As we can see, the results show that there is no buffer space overflow at all ranges, with the exception of range A. This is mainly due to the earlier arrival of MMUs than we expect. Furthermore, MH overcomes the overflow only after it gets a couple of buffer overflow. This basically mirrors the results obtained in Figure 4.10.

In all of the experiments, a small number of MMUs overflow and underflow have obtained using the the bounded jitter case. Therefore, the scheme supports successfully both intra-stream and inter-stream synchronizations with a reasonable and acceptable time delay.

In the next set of experiments, a non-uniform MMU arrival case is assumed. The results obtained are summarized in Figures 4.12 and 4.13. Figure 4.12 displays the values obtained for the buffer space underflow rate as the number of requests are varied, and each request gets four frames from servers. The results indicate that in all cases buffer space is underflow. We also observe different values of the buffer space underflow rate. This is due to the size of the different values of the jitter. As we can see, the buffer space underflow rate increases as as the the value of the average jitter is increased This is due to the lack of MMUs that could be saved at mobile host. We also notice, that space underflow decreases as the number of requests is increased, the value of the buffer BSs accumulate information about delay time, and in time MHs

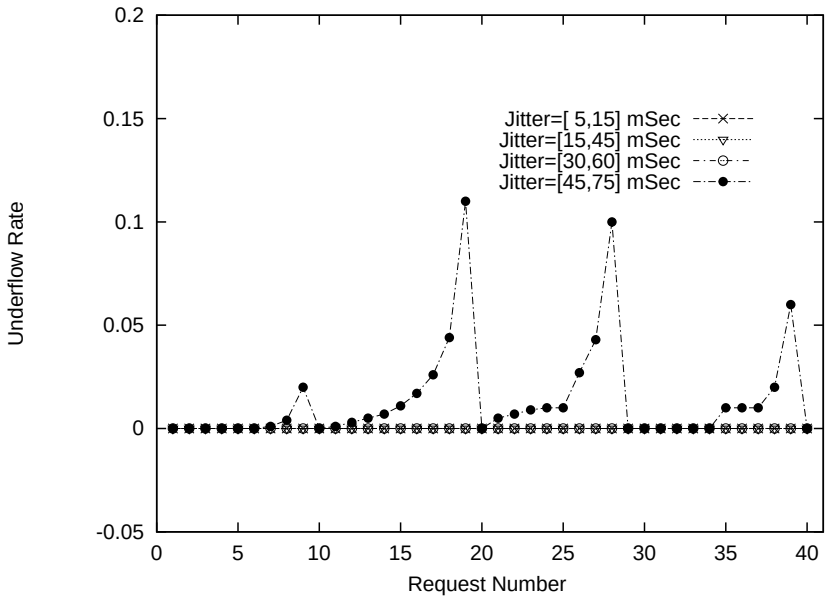

Figure 4.10: Underflow Vs MMU Request number:Uniform

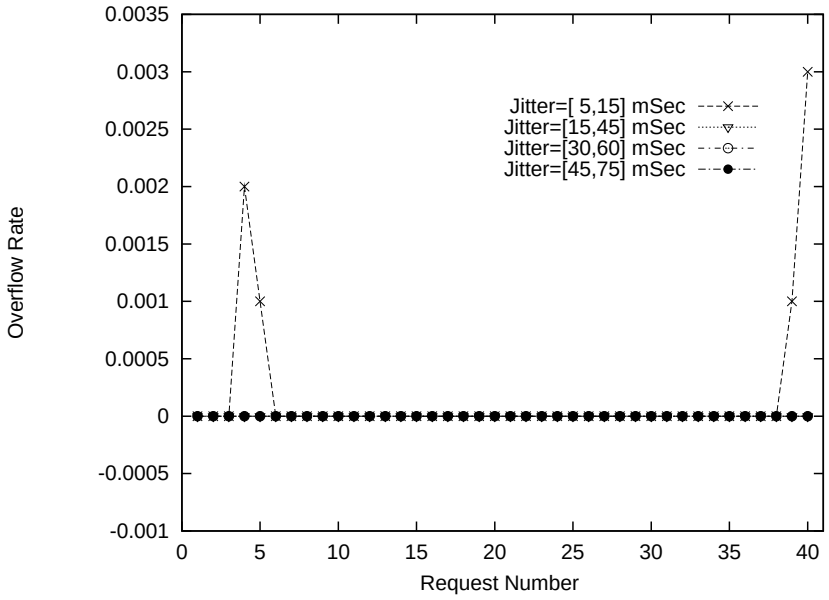

Figure 4.11: Overflow rate Vs MMU Request number:Uniform case

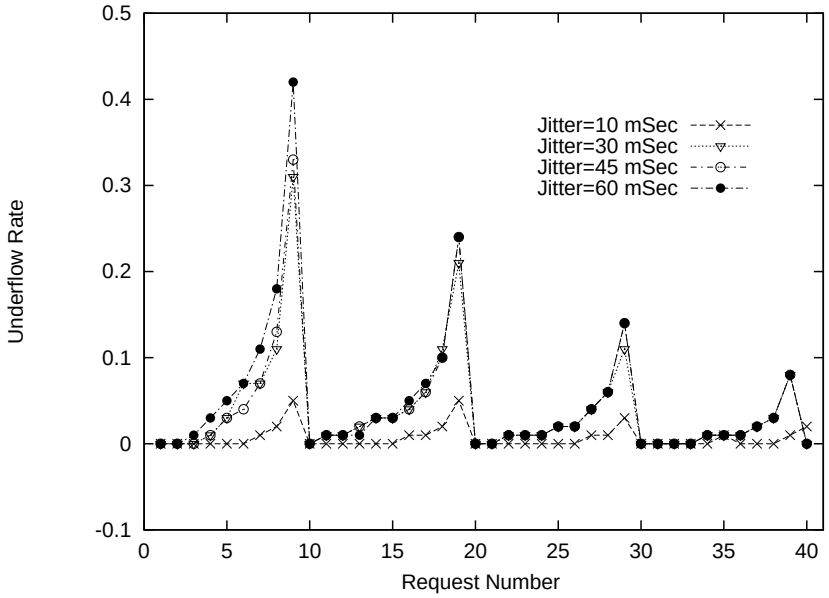

Figure 4.12: Underflow of messages Vs MMU Request number:Non-Uniform

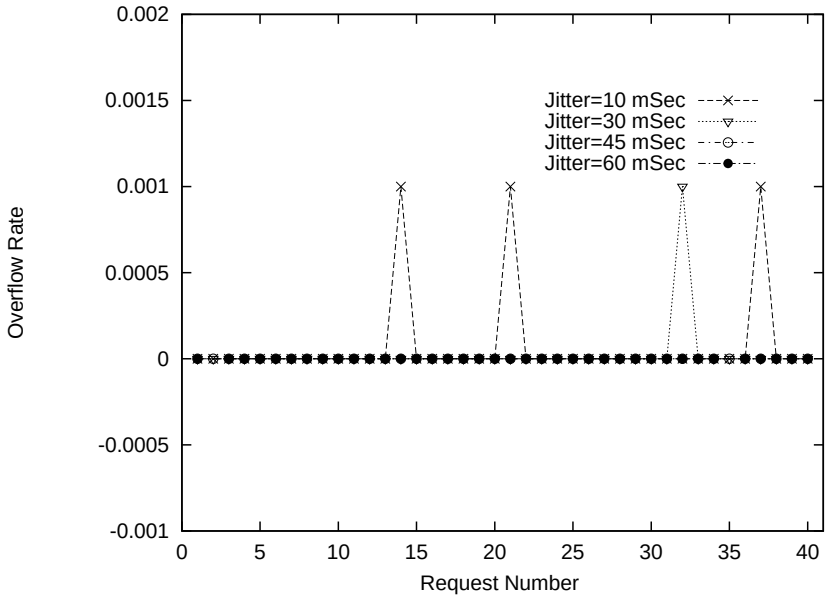

Figure 4.13: Overflow Vs MMU Request number:Non-Uniform case

will get the accurate buffer size from the BSs . This is one of the main reasons of the reduction of the buffer space underflow rate reduced while increasing the number of requests.

Figure 4.12 displays the values obtained for the buffer space overflow rate as the number of requests are varied. As we can observe, the buffer space rate is under 1% when ranges A and B are used. However, we notice no buffer space overflow when the ranges C and D are used. These results are very promising.

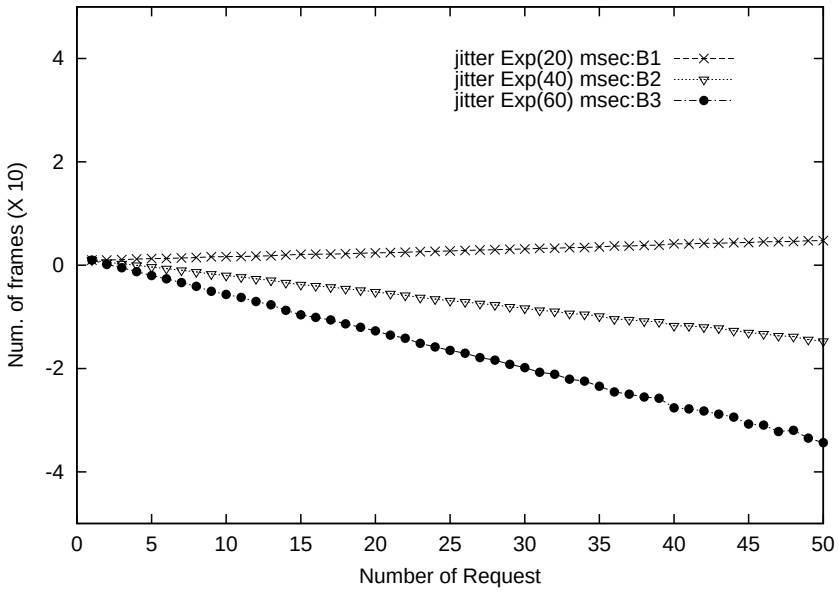

Figure 4.14: Buffer usage:Non-Uniform

• Part B

Figures 4.14-4.15 portray the buffer usage of MoSync according to the number of multimedia requests. As we can see in Figure 4.14, range A_1 needs a buffer size that is 18 times that of an MMU while range A_2 needs a buffer size only 10 times of an MMU, both at the 50 request maximum. Range A_3 requires about 2 MMUs more to play-out smoothly at the same number of requests, while range A_4 needs 12 MMUs. Figure 4.14 shows the buffer usage of the non-uniform case in Part B. Buffer usage with the ranges A_2 and A_3 is decreasing linearly and, at 50 requests, usage with the

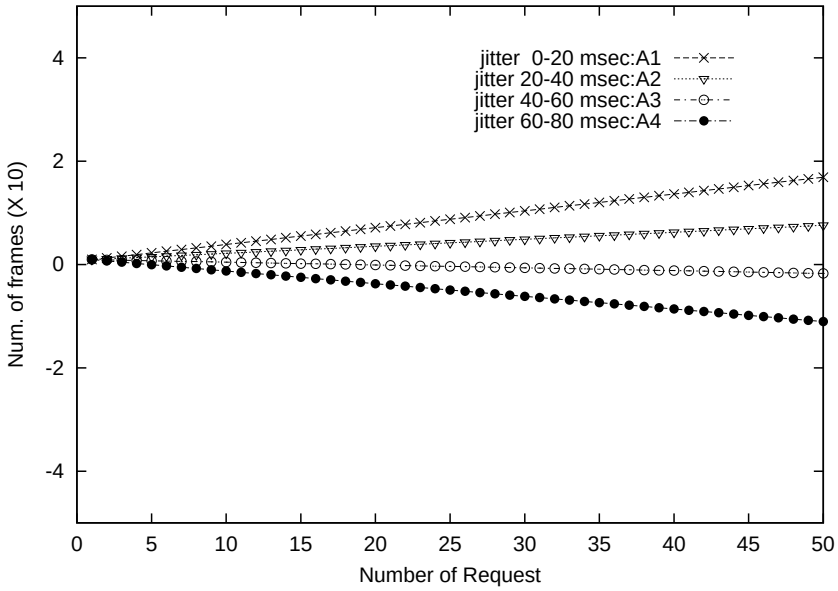

Figure 4.15: Buffer usage:Uniform

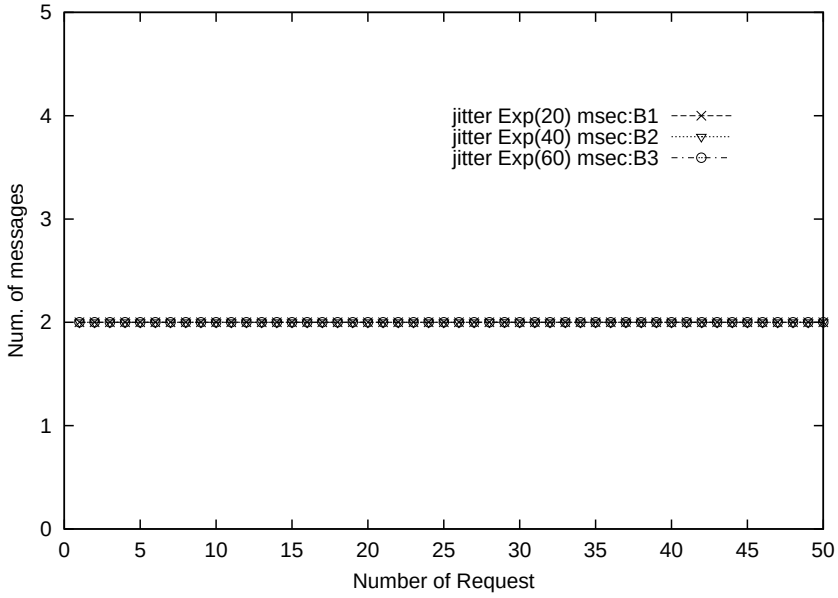

Figure 4.16: Message Complexity:Non-Uniform

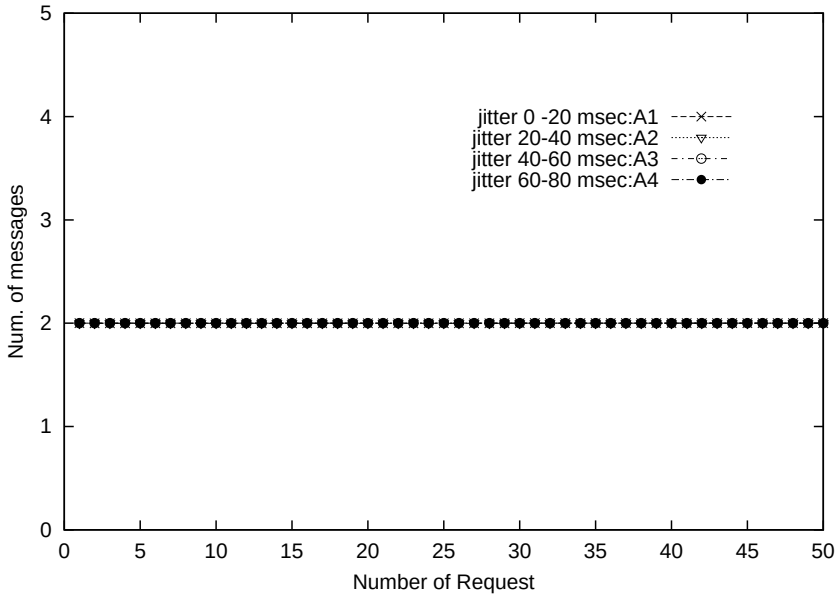

Figure 4.17: Message Complexity:Uniform

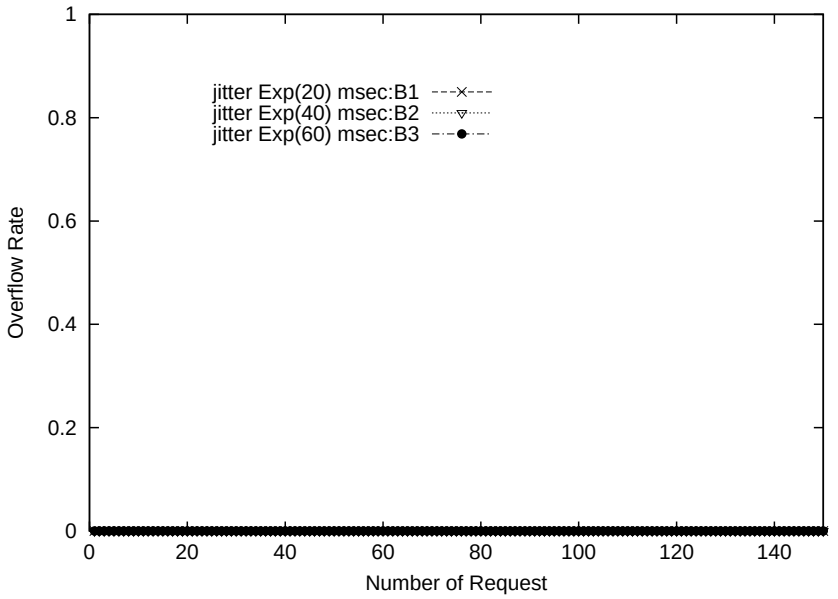

Figure 4.18: Overflow Rate:Non-Uniform

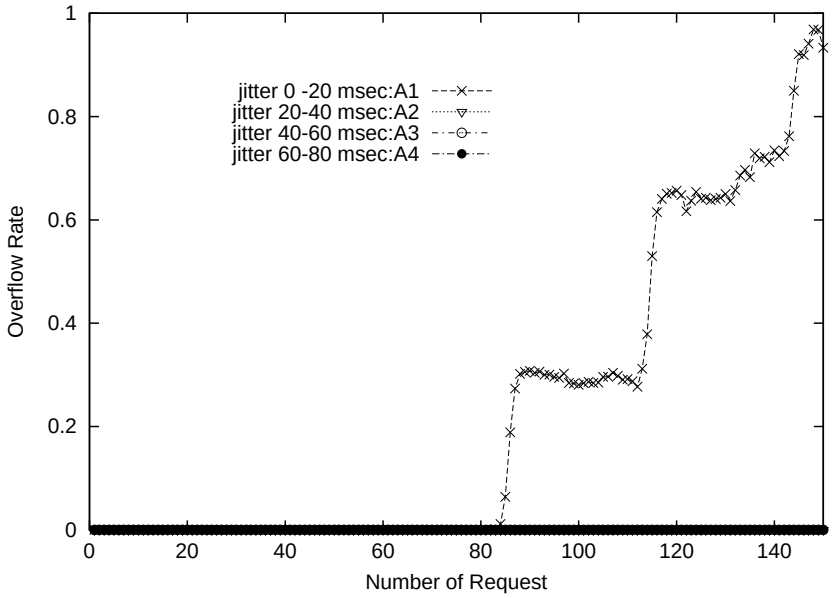

Figure 4.19: Overflow Rate:Uniform

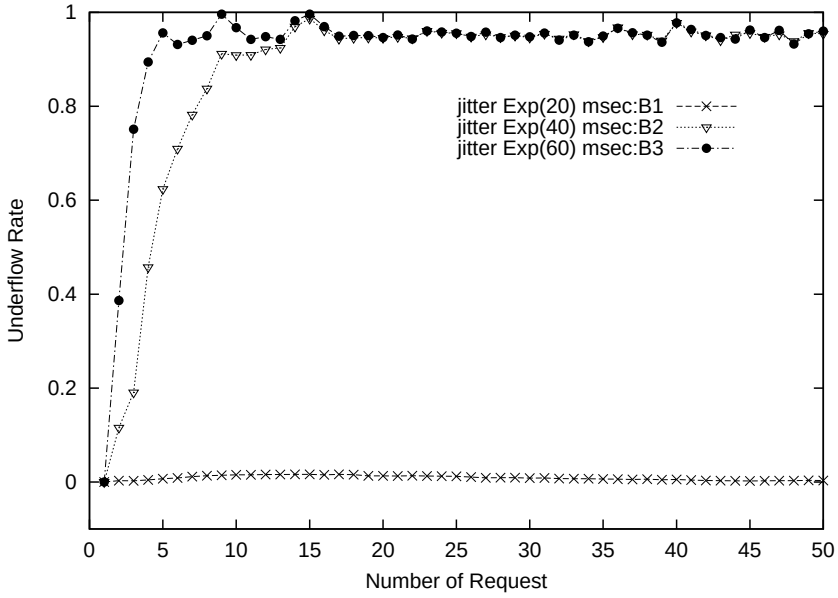

Figure 4.20: Underflow Rate:Non-Uniform

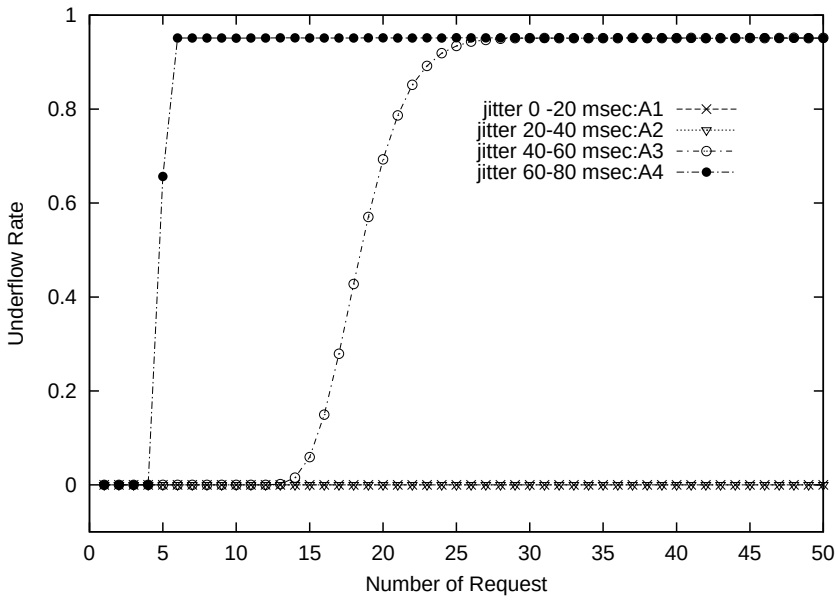

Figure 4.21: Underflow Rate:Uniform

range A_4 requires around 38 MMUs to play-out smoothly. At 50 requests, he number of frames needed for range A_1 is grater than 0 and smaller than 10. Figures 4.16-4.17 portray the message overhead of the algorithm for both the uniform and non-uniform cases. As we can see, message complexity is almost a constant 2, because MHs share information about arrival times. Figures 4.18-4.19 portray the overflow rate for both the uniform and non-uniform case. As we can see the scheme performs quite well in all ranges of jitter, except for range A_1 in the uniform case. In Figure 4.19, buffer usage for range A_1 increases dramatically only after 60 requests, because there is no feedback control or buffer control function in Part B. We observe that the low jitter causes overflow without feedback control. In Figures 4.20-4.21 a high rate of underflow is produced, except with ranges A_1 and A_2B in the uniform case and range A_1 in the non-uniform case. In Figure 4.20, range A_1 shows almost no underflow at all, but ranges A_2 and A_3 show 100% underflow after 50 requests. In Figure 4.21, the results indicates that the data in ranges A_1 and A_2 do not cause underflow. In the simulations, the rate of underflow for the ranges A_3 and A_4 reaches 100% at 50

requests.

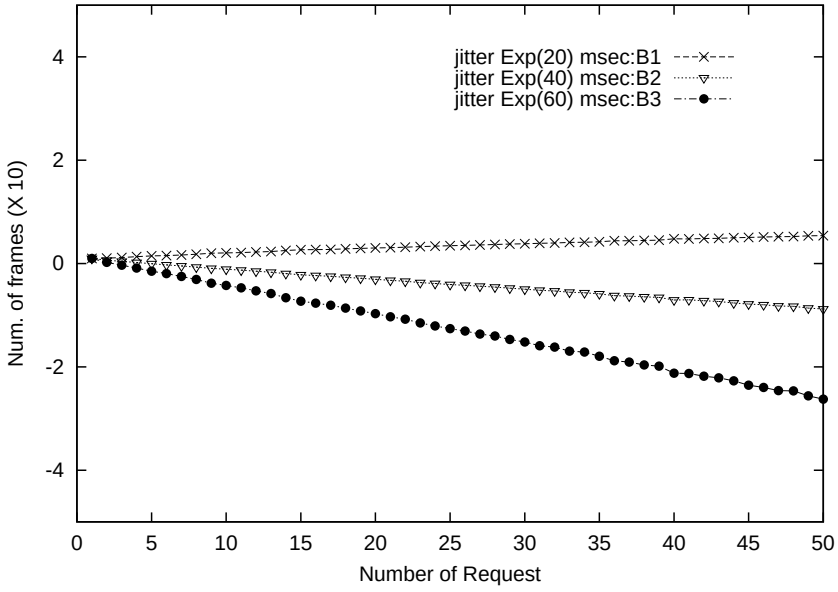

Figure 4.22: Buffer usage:Non-Uniform

• **Part C**

In the experiments, a feedback mechanism to control buffer usage in each MH is used. We also continually update the delivery time information at BS for MHs located on its cell. The feedback mechanism here is to find the expected arrival time for the next MMUs. The equation $E_t = (1 - \alpha)N_t + \alpha OE_t$ is used to get the next arrival times for the next MMUS, where α is fixed at 0.5, E_t is the expected arrival time, N_t the current arrival time, and OE_t the old arrival time. After MoSync calculates E_t, it considers current buffer usage, deciding on a new synchronization time in order to speed up or delay new arrival times, as needed. Delaying new arrivals is much easier than speeding them up because arrival time is highly dependent on network and server conditions. Note that E_t is only an expected arrival time, not a guaranteed delivery time for the next MMUs. Establishing such synchronization points, allows the number of messages to be reduced, because the base station is able to combine

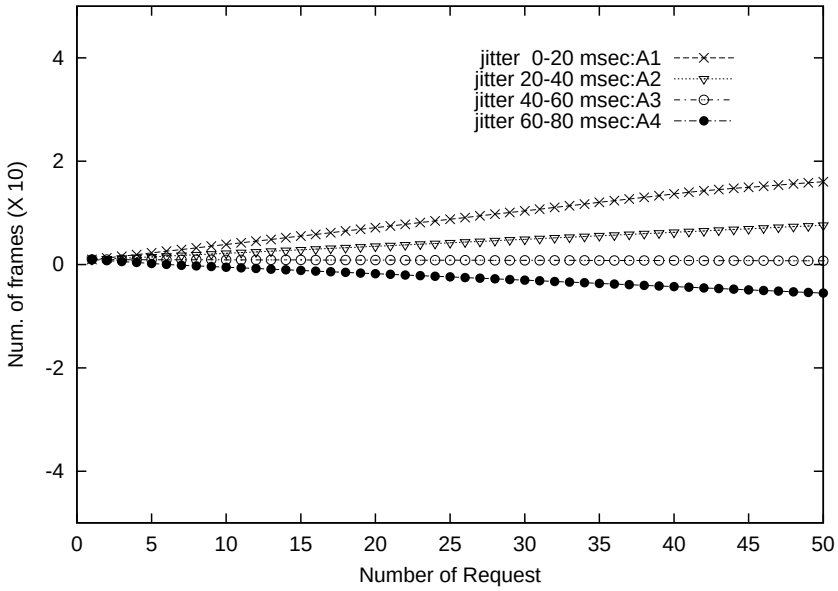

Figure 4.23: Buffer usage:Uniform

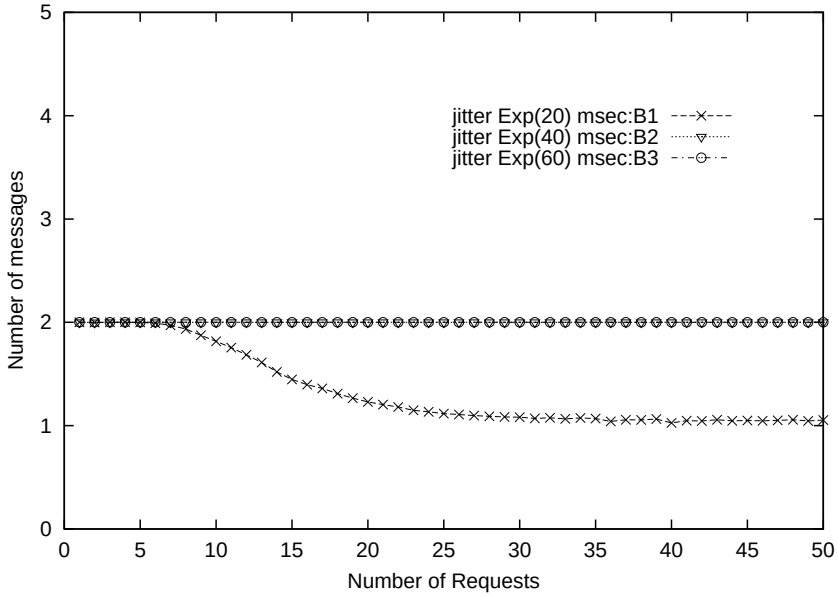

Figure 4.24: Message Complexity:Non-Uniform

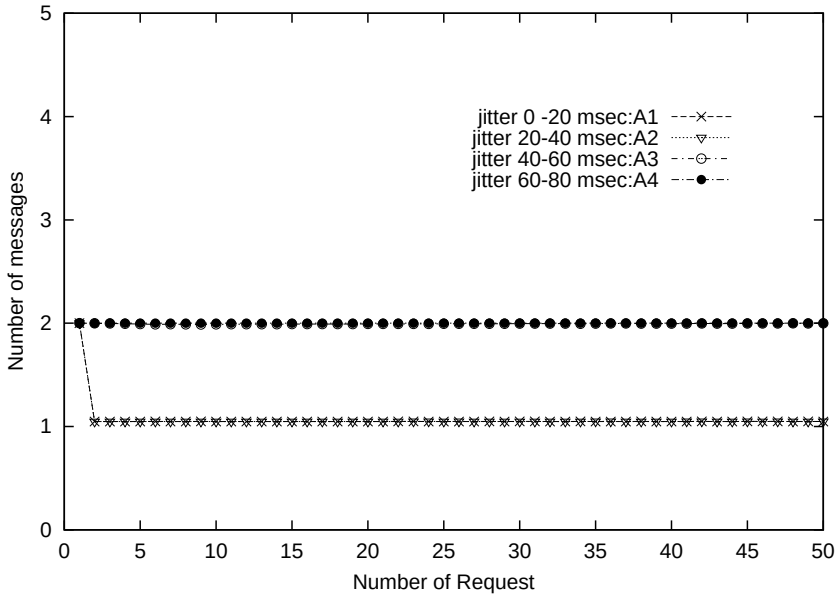

Figure 4.25: Message Complexity:Uniform

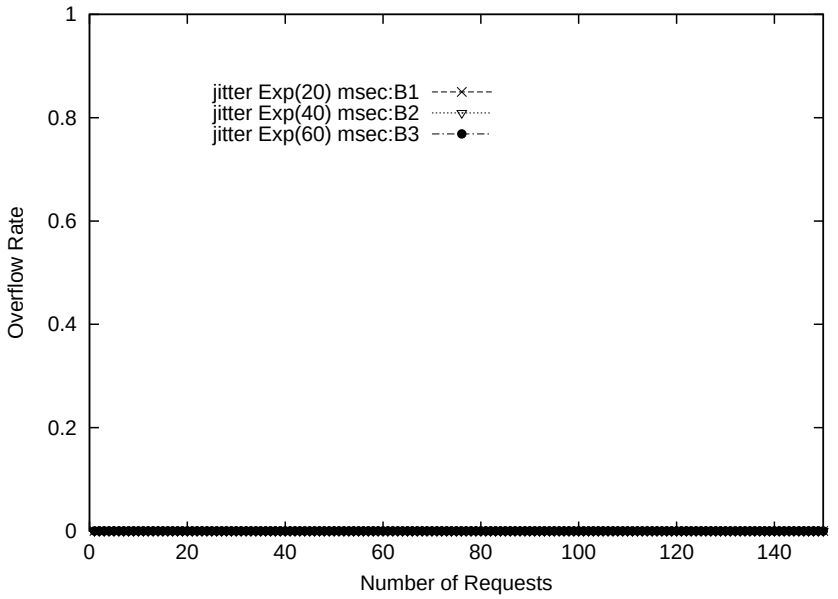

Figure 4.26: Overflow Rate:Non-Uniform

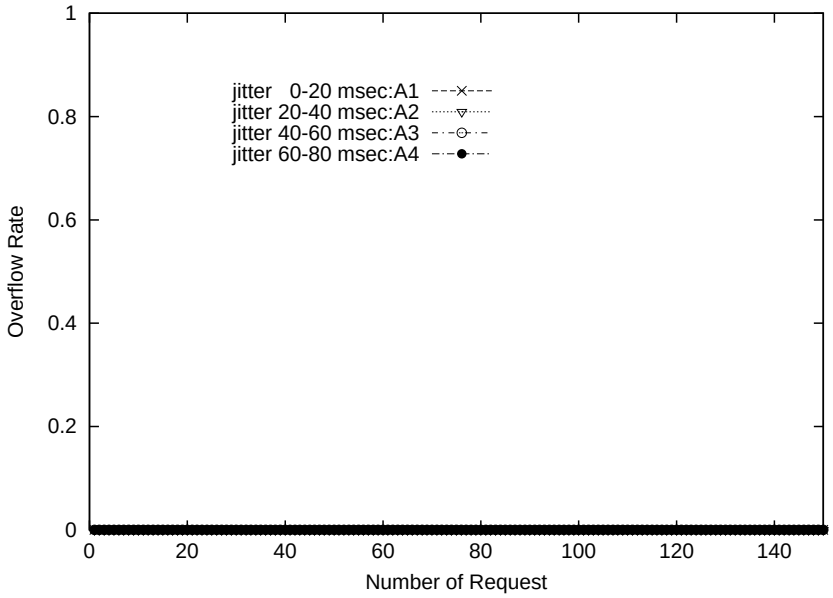

Figure 4.27: Overflow Rate:Uniform

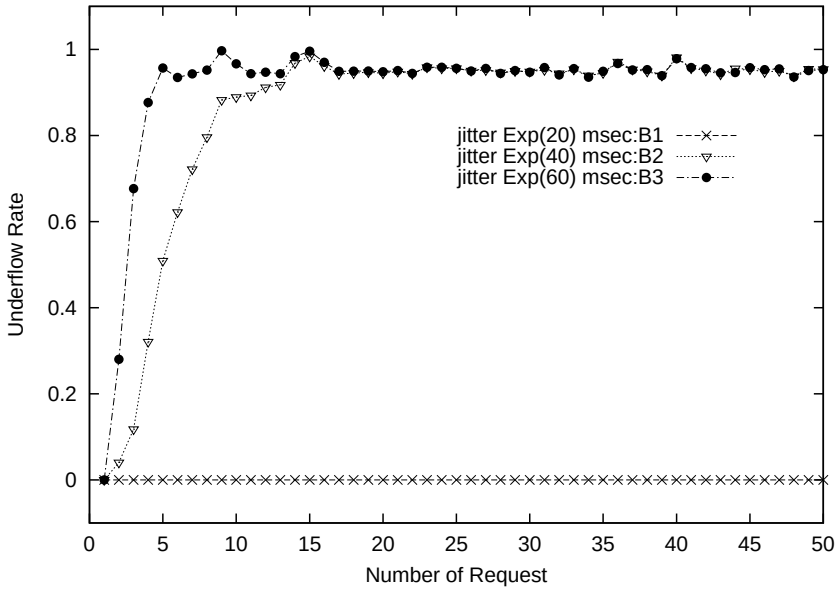

Figure 4.28: Underflow Rate:Non-Uniform

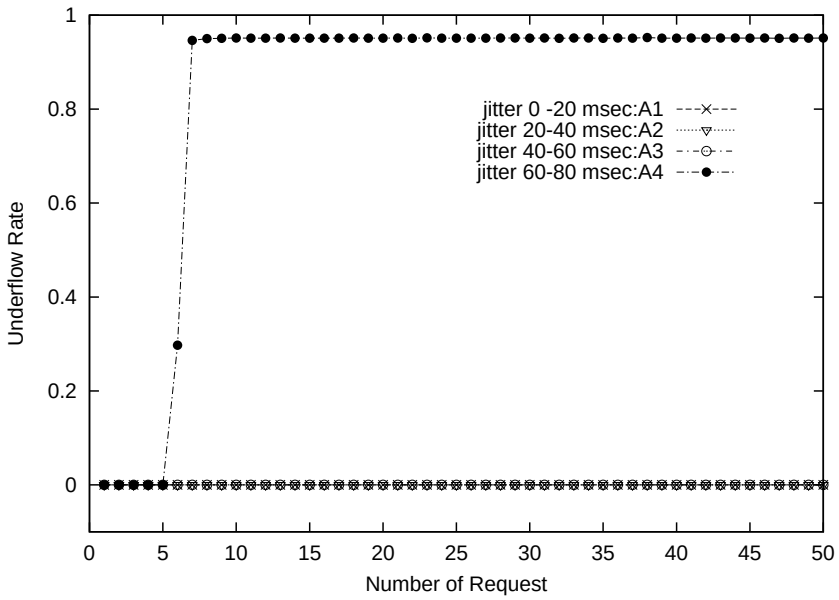

Figure 4.29: Underflow Rate:Uniform

the synchronous information from all its participating MHs, to establish a uniform rate at which future date will be sent without needing additional request messages from the MHs. Here the base stations and servers can choose independently to use either pessimistic or optimistic delivery.

The results obtained for buffer usage are shown in Figures 4.22-4.23 in the uniform and non-uniform arrival cases. In Figure 4.22, buffer usage for the ranges B_1, B_2 and B_3 is always greater than 0 and less than 18 MMUs, which means that all MMUs can continue to play-out smoothly over the full range of 50 requests. With range B_4, underflow occurs before the request is received. The data for range B_4 is 10% in Part C. Note that the rate of claimed buffer space is the number of frames divided by the the number of requests.

For example, with range B_4, a MH needs buffer space for 5 MMUs on average over the full 50 requests. The result is a 10% drop rate for the MMUs. In the Figure 4.23, the data of range B_3 claims more buffer space.

Figures 4.24-4.25 show that the scheme performs quite well in all ranges of jitter,

because the average number of messages per MMU is close to the constant 2. In the special case of ranges B_1 and B_2, message complexity is close to 1 after two MMU requests in the uniform case. In the non-uniform case, message complexity for range B_1 is essentially a constant 1 after ten MMU requests.

Figures 4.26-4.27 portray the overflow rate for both uniform and non-uniform case. As we can see the scheme performs very well in all ranges of jitter. The algorithm removes all overflow up to the maximum 150 MMU requests.

Figures 4.28-4.29 show the results of the schemes in the case of Figures 4.29 show the results of the schemes in the case of underflow. In Figure 4.28, only the data for range B_4 produces underflow after 5 MMU requests. Also, almost 95receive enough MMUs to play-out smoothly. The other three ranges B_1,B_2, and B_3 do not produce any underflow. In range B_3, the results show that MoSync causes a dramatically low underflow rate. In Figure 4.29, the underflow rate for range B_1 is essentially 0, which means there is no underflow. Based upon these results above, MoSync in the case of bounded reasonable jitter supports inter-stream synchronization with an acceptable delay. Note that MoSync uses a delay time to allow intra-synchronization and thereby guarantees inter-stream synchronization as well.

4.6 Conclusion

New technologies in multimedia and wireless telecommunication have initiated many research topics in wireless and mobile multimedia systems. Although the synchronization problem for distributed multimedia systems has been going on for the last decade, it seems little work was done in a wireless multimedia environment. In this chapter, an algorithm for both inter- and intra-stream synchronizations of stored multimedia streams has been presented. the scheme ensures the synchronization among mobile clients and multimedia units has been shown with different delays, jitter, server drop-outs, and clock drift. The scheme is discussed and its proof of correctness is showed. The results obtained from experiment to evaluate its performance using no jitter case with message complexity and buffer usage are presented. The algorithm's performance is reported with several jitter cases using number of underflow and overflow

at each frames arrival time. The scheme exhibits a better message complexity when compared to an enhanced sink-source scheme and adapted to a wireless environment.

Symbols	Meaning
L	number of Mobile clients
M	Number of base stations
K	number of servers
I	number of media
k	server index k=1,. . . ,K
i	media index i=1,...,I
l	mobile client index l=1,...,L
m	base station index m=1,...,M
n	base station $n \neq m$ and n=1,...M
δ	minimum delay difference
S_k	a server k producing substream k
BS_m	a Base station m;
BS_n	a base station n;
MH_l	a wireless multimedia client
MM	media unit consisting n streams
MS_k^i	multimedia substream i
DS_l	play-out rate for MH_l
TS_k^i	Time stamp for server k
TS_l^i	Time stamp for mobile host l
DT_k^0	delay time for server k
ST_k^0	arrival time from server k
MP^i	middle point (time) for buffer
CU	Current mobile clients
RT_k^i	round time from MH_l to S_k
NE_ON	a neighboring BS is acting
NE_m	neighboring base station of BS_m

Table 4.1: Notations

$$\begin{aligned} buf_{all} &= \sum_{k=1}^{n} buf_k \\ &= \sum_{k=1}^{n} (2\,(d_k^{max} - d_k^{min})\,r) \\ &\leq N\,2\,\delta^{max}\,r \end{aligned}$$

Number of cells	60
Number of Multimedia Servers	4
size of buffer of a MH	30 times of a MMU
Ployout time/MMU	100mSec
Mean service time /session	μ
Arrival rate in normal cell	λ
RTT to request/deliver a MMU	50 mSec
jitter(Uniform)	A_1[0-20],A_2[20-40],A_3[40-60],A_4[60-80]mSec
jitter(Non-Uniform)	B_1[0-200], B_2[0-400], B_3[0-600] mSec

Table 4.2: Simulation Parameters

CHAPTER 5

CONCLUSION

No area of telecommunication is growing faster than wireless and mobile networking. With the advances in telecommunications over the past decade, computer users now enjoy virtually unlimited access to any kind of information they need. However, mobility, hand-in-hand with communication, opens up a multitude of new issues and trade-offs. The dynamic re-reconfigurability of wireless networks provokes challenges in many areas including power control, resource allocation/management, and location management. Wireless telecommunication technologies make wireless multimedia systems feasible. Although the synchronization problem for distributed multimedia systems has received active research attention for the last decade, little work has been done in a wireless multimedia environment for PCS. To make wireless multimedia system using PCS, resource allocation problems and multimedia units synchronization problems have been studied.

In this chapter, a new contribution to the resource allocation management and multimedia unit synchronization management are summarized. At first, the contributions of dynamic channel assignment for wireless communication systems making use of the mutual exclusion model are addressed where the channels are grouped according to the number of cells into groups and each group of channels can not be shared concurrently by neighboring cells.

- an efficient distributed algorithm for dynamic channel allocation is presented. the algorithm requires less communication overhead than previous schemes, $O(N_g * N_n log N_n)$ messages Versus $O(N_g^2 * N_n)$ for Choy's algorithm, where N_g is the number of channel groups, and N_c is the number of channels in a system.

- a simulator has been developed from scratch to simulate the algorithm and others. Its implementation, and presented the results obtained to evaluate its performance using several channel systems and different types of call arrival patterns have been discussed. The results indicate that a 400 channel system

produces the best results in all call arrival patterns (i.e., uniform, non-uniform even, and non-uniform random). A low denial rate, and low message complexity have been obtained in most of the experiments. Extensive simulation experiments were conducted to compare the performance of the algorithm with previous wireless channel allocation algorithms. The experimental results indicate that a significant reduction of the denial rate were obtained using the algorithm when compared to the use of optimistic/pessimistic schemes. For instance, a 5% denial rate improvement over previous algorithms was observed when a system load of 100 is used and a non-uniform distribution model. The results obtained also indicate that the acquisition time using the DDRA algorithm and a non-uniform model with 100 system load, is at about 35-45% faster than previous algorithms. Last, but not least, the DDRA algorithm exhibits an upper bound of 400 mSec when a non-uniform model is used, where previous algorithms exceeded this when the system load is increased.

Second, MoSync, a Quasi-sink algorithm for both inter- and intra-stream synchronizations of stored multimedia streams has been presented. MoSync ensures synchronization among mobile clients and multimedia units under a wide range of delay, jitter, server drop-out, and clock drift is showed.

5.0.1 Primary Research Results

- developed a scheme that ensures the synchronization among mobile clients and multimedia units with different delays, jitter, server drop-outs, and clock drift. the scheme is discussed and its proof of correctness is showed. The results obtained from experiment are presented to evaluate its performance using no jitter case with message complexity and buffer usage

- the algorithm's performance copes with limited jitter cases at each frames arrival time in terms of number of underflow and overflow.

- proved that the scheme exhibits a better message complexity, when compared to an enhanced sink-source scheme and adapted to a wireless environment.

- developed MoSync Simulator from scratch.

5.0.2 Directions for Future Research

- currently investigating modifying the synchronization scheme for soft handoff of CDMA and GSM with macro diversity [23].
- extension of DDRA to dynamic bandwidth assignment could be investigated.
- MoSync and DDRA for dynamic bandwidth could be developed for verietized wireless multimedia systems.

BIBLIOGRAPHY

[1] N. Agra & S. Son *Synchronization of distributed multimedia data in an application-specific manner* Proceedings of the second ACM international conference on multimedia '94, ACM, 1994 pp 141-148.

[2] P. Alleyne and C. Tropper, *The parallel simulation of distributed channel allocation algorithms*, Special issue on Modeling Analysis, Simulation of Wireless Systems Eds. A. Boukerche and H. Jiang, ACM/Baltzer MONET, Vol. 5, No. 3, 2000.

[3] N. Alon and M. Tarsi, *Colorings and orientations of graphs*, Combinatorica, vol. 12 no. 2, 1992, pp.125-134.

[4] B. Awerbuch and M. Saks, *A dining philosophers algorithm with polynomial response time,* Proc. 81st IEEE Symposium on Foundations of Computer Science, St. Louis, MO, Oct. 1990, pp. 65-74.

[5] B. R. Badrinath, A. Acharya, and T. Imielinski, *Impact of mobility on distributed computations*, ACM Operating Systems Review, vol. 27, no. 2, 1993, pp15-20.

[6] J. Bar-han and D. Peleg, *Distributed resource allocation algorithms,* Proc. 6th International Workshop on Distributed Algorithms, 1992, pp. 277-291.

[7] E. Biersack & W. Geyer, *Synchronization delivery and play-out of distributed stored multimedia streams* ACM/Springer-Berlag: Multimedia Systems, Vol. 7, 1999, pp. 70-90.

[8] G. Blakowski & R. Steinmetz, *A media synchronization survey: reference model, specification, and case studies* IEEE Journal on selected areas in communications, Vol. 14, No 1, Jan 1996, pp 5-35.

[9] U. Black, *Mobile and Wireless Networks* Prentice Hall, New Jersey. 1996

[10] A. Boukerche, S. Hong & T. Jacob, *A distributed channel allocation algorithm for mobile communication* IEEE Eighth International Symposium on Modeling,

Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2000), Aug. 29- Set. 1, San Francisco, CA, USA 2000.

[11] G. Cao and M. Singhal *Distributed fault-tolerant channel allocation for mobile cellular networks,* IEEE INFOCOM'99, Mar. 1999, pp. 584-591.

[12] K. Chandy and J. Misra, *The drinking philosophers problem,* ACM Transactions on Programming Languages and Systems, vol. 6, 1984, pp. 632-646.

[13] M. Choy and A. K. Singh, *Efficient fault-tolerant algorithms for distributed resource allocation,* ACM Trans. on Programming Language and Systems, vol 17r, no 3, 1995, pp. 535-559.

[14] M. Choy and A. K. Singh, *Efficient distributed algorithm for dynamic channel assignment,* Proceedings of 7th IEEE Intl' Symp. on Personal, Indoor and Mobile Radio Communication, 1996.

[15] J. P. Coutiat & R. C. Oliveria, *Proving temporal consistency in a new multimedia synchronization model* Proc. of ACM Multimedia, Vol. 4, 1996, pp. 141-152.

[16] S. K. Das, S. K. Sen and R. Jayaram *A dynamic load balancing strategy for channel assignment using selective borrowing in cellular mobile environment,* ACM Wireless Networks, vol. 3, no 5, 1997, pp. 333-347.

[17] S. K. Das, S. K. Sen and R. Jayaram *A novel load balancing scheme for the tele-traffic hot spot problem in cellular networks,* ACM Wireless Networks, vol. 4, no. 4, 1998, pp. 325-340,.

[18] J . W. Dijkstra, *Hierarchical ordering of sequential processes,* Ada Inform atica, vol. 1, fasc. 2,1971, pp. 115-138.

[19] V. H. Mac Donald, *Advanced Mobile Phone Service: The Cellular Concept*, The Bell System Tech., vol. 58, no. 1, Jan. 1979 pp. 15-41.

[20] J. Escobar, C. Partridge & D. Deutsch *Flow synchronization protocol* IEEE/ACM Transaction on Networking, Vol. 2, No 2. , April 1994, pp111-121.

[21] S. Floyd & V. Jacobson, *The synchronization of periodic routing messages* IEEE/ACM Transaction on Networking, Vol. 2, No. 2, 1994, pp 122-136.

[22] G. H. Forman and J. Zahorjan, *The challenges of mobile computing*, Tech. Report, Comp. Science and Eng., Univ. of Washington, 1994.

[23] V. K. Garg, *IS-95 CDMA and cdma2000* Prentice Hall, New Jersey, 1999,

[24] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W. H. Freeman and Company, New York, 1979.

[25] N. D. Georganas, R. Steinmetz & T. Nakagawa, *Synchronization issues in multimedia communication* IEEE Journal on selected areas in communications, Vol. 14, No 1, Jan 1996, pp 1-3.

[26] P. Goyal & H. M. Vin, *On the effectiveness of buffer in deterministic services* Proc. of 8th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '98) July, 1998

[27] T. Imielinski & B. R. Basrinath, *Data management for mobile computing* SIGMOD Record, Vol. 22, No. 1, March, 1993.

[28] Y. Ishibashi, S. Tasaka, and T. Takeo, *A perfrmance comparison of media synchronization schemes for collaborative systems in an interconnected ATM-wireless LAN*, in Proc., IEEE PIMRC'98 pp265–271, 1998.

[29] Y. Ishibashi, and. Tasaka, *A comparative survey of synchronization algorithms for continuous media in network environments* in Proc. IEEE LCN 2000, pp.337-348, Nov., 2000.

[30] J. P. Jarmasz & N. D. Georganas *Designing a distributed multimedia synchronization scheduler* Proc. IEEE International Conference on Multimedia Computing and Systems, Ottawa, Ont. , Canada, June. 1997

[31] T. V. Johnson & A. Zhang, *Dynamic play-put scheduling algorithms for continuous multimedia streams* ACM/Springer-Berlag: Multimedia Systems, Vol. 7, 1999, pp. 313-325.

[32] M. Kato, Y. Kawai, and S. Tasaka *Performance evaluation of media synchronization in PHS with the H.223 Annes multiplexing protocol* in Proc. IEEE ICUPC 1998 pp183-190.

[33] M. Kato, S. Tasaka and K. Nakamura *A comparison of media synchronization quality between TCP and UDP in PHS Internet access* in Proc. IEEE PIMRC 1999 ppc1092-1097.

[34] L. Lamont, L. Li, R. Brimont & N. D. Georganas, *Synchronization of multimedia data for a multimedia news-on-demand application* IEEE Journal on Selected Areas in Communications, Vol. 14, No. 1,1996, pp 264-278.

[35] L. Lamport, *Time, clocks and the ordering of events in a distributed system,* Communication of the ACM, vol. 21, no. 7, pp. 558-565, 1978.

[36] W. C-Y. Lee, *Mobile cellular telecommunications,* 2nd ed., McGraw-Hill, Inc., 1995.

[37] T. D. C. Little, A. Ghafoor, C. Y. R. Chen, C. S. Chang & P. B. Berra, *Multimedia Synchronization* IEEE Data Engineering Bulletin, Vol. 14, No 3, Sept. , 1991,pp. 26-35.

[38] T. D. C. Little & F. Kao, *An intermedia skew control system for multimedia data presentation* Proc. 3rd Intl. , Workshop on Network and Operating System Support for Digital Audio and Video, La Jolla, CA, Nov. , 1992,pp. 121-132.

[39] N. Lynch, *Upper bounds for static resource allocation in a distributed system,* Journal of Computer and System Science, vol. 23, 1981, pp. 254-278.

[40] P. Tsigas, N. Garg, M. Papatriantafilou, *Distributed list coloring: How to dynamically allocate,* Tech. report, Max-Planck Institute for Computer Science, Saarbrucken, Germany, 1996.

[41] C. Nicolaou, *An architecture for real-time multimedia communication systems*, IEEE J. Selected Areas in Comm. , Vol. 8, No. 3, April 1990, pp. 391-400.

[42] H. Okura, M. kato, and S. Tasaka *A media synchronization experiment on continuous media transmission in Bluetooth LAN access* in Proc. IEEE PIMRC 2001.

[43] I. Page, R. Jacob and S. Chern, *Fast algorithms for distributed resource allocation,* IEEE Transactions on Parallel and Distributed Systems, Feb, 1993, pp. 632-646.

[44] M. J. Perez-Luque & T. D. C. Little, *A temporal reference framework for multimedia synchronization* IEEE Journal on Selected Areas in Communications, Vol. 14 No. 1, 1996, pp36-51.

[45] R. Prakash, N. G. Shivarati, and M. Singhal, *Distributed dynamic fault-tolerant channel allocation for cellular networks,* Proceedings of 14th ACM Symp. on Principles Dist. Computing, 1995, pp. 47-56.

[46] N. U. Qazi, M. Woo & A. Ghafoor, *A synchronization and communication model for distributed multimedia objects* Proceedings of the conference on Multimedia '93, ACM, 1993, pp 147-155.

[47] S. Pamanathan & V. Rangan, *Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks* IEEE/ACM Transactions on Networking. Vol. 1, No. 2, April 1993, pp 246-260.

[48] E. Styer and G. Peterson, *Improved algorithms for distributed resource allocation,* Proc. 7th ACM Symposium on Principles of Distributed Computing, Toronto, Canada, August 1988, pp. 105-116.

[49] S. Tasaka, Y. Ishibashi, and H. Imura, *Stored media synchronization in wireless LANs* in Conf. Rec IEEE GLOBECOM '96 pp 1904-1910.

[50] S. Tasaka, and Y. Ishibashi, *Stored media synchronization schemes in ATM and wireless networks: A performance comparison* in Conf. Rec IEEE ICUPC'97, pp 766-772 1997.

[51] S. Tasaka, H. Nakanishi and Y. Ishibashi, *Dynamic resolution control and media synchronization of MPEG in wireless LANs* in Conf. Rec IEEE GLOBECOM '97 pp 138-144.

[52] A. Thesen and L. E. Travis, *Simulation for decision making,* West Publishing Co., 1992.

[53] M. Townsend, *Discrete mathematics: applied combinatorics and graph theory,* Benjamin/Cummings Publishing Co., 1987.

[54] J. Welch and N. Lynch, *A modular drinking philosophers algorithm,* Distributed Computing, vol. 6, 1993, pp. 233-244.

[55] E. Weidman, I. Page, W. Pervin, *Explicit dynamic exclusion algorithm,* Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing, December 1991, pp. 142-149.

[56] K. -L. Wu & P. H. Yu, *Consumption-based buffer management for maximizing system throughputs of news-on-demand multimedia systems*

[57] T. Znati, R. Simon & B. Field, *A network-based scheme for synchronization of multimedia streams* TR95-02, Dept. of Computer Science, Univ. of Pitt. , Pitt. , PA. 1995