

ADAPTIVE HYBRID HANDOVER SCHEME OVER ATM-BASED PERSONAL COMMUNICATION NETWORKS

A DISSERTATION

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

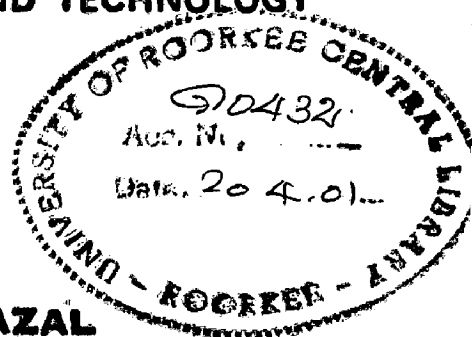
MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND TECHNOLOGY

By

SAIB M. GAZAL



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)**

FEBRUARY, 2001

CANDIDATE'S DECLARATION

I hereby certify that the work which is presented in this dissertation entitled "ADAPTIVE HYBRID HANDOVER SCHEME OVER ATM-BASED PERSONAL COMMUNICATION NETWORKS", in partial fulfilment of the requirements for the award of degree of MASTER OF TECHNOLOGY (M.Tech.) in COMPUTER SCIENCE AND TECHNOLOGY (CST), submitted in the Department of Electronics and Computer Engineering, University of Roorkee, Roorkee, is an authentic record of my own original work carried out from August 2000 to February 2001 under guidance of **Dr. Mohan Lal**, Assistant Professor, University Computer Center, University of Roorkee, Roorkee and **Dr. A.K. Sarje**, Professor, Department of Electronics and Computer Engineering, University of Roorkee, Roorkee.

The matter embodied in this dissertation has not been submitted by me for the award of any other degree or diploma.

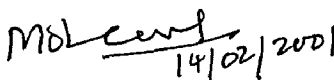
Date: 14th Feb 2001

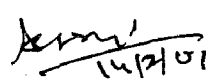
Place: Roorkee


(SAIB M. GAZAL)

CERTIFICATE

This to certify that the above statement made by the candidate is correct to the best of our knowledge and belief.


(Dr. Mohan Lal)
Assistant Professor
University Computer Center
University of Roorkee
Roorkee - 247 667
INDIA


(Dr. A.K. Sarje)
Professor
E&C Department
University of Roorkee
Roorkee - 247 667
INDIA

ACKNOWLEDGMENT

It gives me a great pleasure to take this opportunity to thank and express my deep sense of gratitude to my guides, Dr. Mohan Lal and Dr. A.K. Sarje, for their valuable suggestions, guidance and constant encouragement during the course of this dissertation work. I deem it my privilege to have carried out this dissertation under their guidance.

I would like to thank all my friends and well wishers, who helped me directly or indirectly in making this report.

Finally, I express my regards to my parents, who have been a constant-source of inspiration to me.


(SAIB M. GAZAL)

ABSTRACT

Personal communications network (PCN) is an emerging wireless network that promises new services for the telecommunication industry. Handover is the action of switching a call in progress in order to maintain the continuity and the required quality of service (QoS) of a call when a mobile terminal (MT) moves from one cell to another. In literature, queuing of handover requests, and reserving number of channels (statically or dynamically) exclusively to serve handover request are proposed. Those schemes aim to reduce the failure probability of handover requests (forced termination probability).

In this dissertation a handover hybrid scheme is proposed, which combines queuing of the handover requests scheme and guard channel (or bandwidth reservation) scheme, which gives a higher access priority to handover calls by assigning them a higher capacity. In this scheme, a new call is admitted only if number of idle channels is greater than fixed number of guard channels (N_G) and there is no handover request exists in the handover queue. While handover request is admitted if any channel is free, otherwise, it is queued in handover queue instead of blocking the request.

This scheme decreases significantly the forced termination probability compared to queuing or bandwidth reservation schemes. The expense is an increase in the new call blocking probability. However, from the subscriber's point of view, forced termination of due to handover is less desirable than blocking new calls.

The proliferation of demand for extending wireless services to integrated services, which support the transmission of data and multimedia information has resulted in need for broadband wireless systems that are able to provide service capabilities similar to those of wireline networks. The ATM cell-relay paradigm is one possible approach to provide switching networks for interconnection of PCN cells.

The implementation is done in C++ language on Intel Celeron personal computer under DOS environment.

CONTENTS

	Page No.
Candidate's Declaration.....	i
Acknowledgement.....	ii
Abstract.....	iii
Chapter- 1: INTRODUCTION.....	1
1.1 ATM-based PCN.....	2
1.2 Mobile ATM.....	2
1.3 Problem Statement.....	3
1.4 Organization of Dissertation.....	4
Chapter- 2: HANDOVER SCHEMES.....	5
2.1 Queuing Handovers.....	5
2.1.1 FIFO Queuing scheme.....	5
2.1.2 Measurement-Based Prioritized Scheme	6
2.1.3 Signal Prediction Priority Queuing.....	8
2.2 Bandwidth reservation based scheme.....	8
2.2.1 Fully Shared Scheme.....	10
2.2.2 Reserved Channel Scheme.....	10
2.2.3 Dynamic Reserved Channel Scheme.....	10
2.3 Subrating Scheme.....	13
2.4 Channel carrying and borrowing scheme.....	13
Chapter- 3: HYBRID HANDOVER SCHEME.....	14
3.1 Admission Control.....	14
3.2 Extension Over ATM-Based Network.....	16

	3.2.1	Admission Control Over backbone Network.....	16
	3.2.2	Backbone Handover.....	19
Chapter-	4:	SIMULATION MODEL.....	21
	4.1	Traffic Model.....	21
	4.2	Simulation Model I.....	22
	4.2.1	Simulation Parameters.....	22
	4.3	Simulation Model II.....	22
	4.3.1	Environment Description.....	23
	4.3.2	Simulation Environment.....	25
	4.3.3	Mobility Model.....	26
	4.3.4	Simulation Parameters.....	27
	4.4	Simulation Program.....	27
Chapter-	5:	RESULTS AND DISCUSSION.....	28
	5.1	Blocking in FSS.....	28
	5.2	Results in Simulation Model I.....	28
	5.2.1	Forced Termination Probability.....	30
	5.2.2	Blocking Probability.....	30
	5.2.3	Improvement.....	32
	5.2.4	Blocking Increase.....	34
	5.2.5	Carried Load.....	36
	5.3	Results in Simulation Model II.....	38
	5.3.1	Reserve Channels at Backbone Link.....	38
	5.3.2	Improvement of Reserving Channels at Backbone Link.....	41
	5.3.3	Increase in Blocking Due to Reserving Channels at Backbone Link.	44
	5.3.4	Carried Traffic.....	44

Chapter-	6:	CONCLUSION	46
	6.1	Scope for Future Work.....	47
REFERENCES			48
APPENDIX: Software listing			50

INTRODUCTION

Cellular systems require handover procedure, as a single cell does not cover the whole service area. The smaller the cell size and the faster the movement of MT through the cells, the more handovers of ongoing calls are required. There are two basic reasons for a handover [1]:

- The mobile station moves out of the range of a base transceiver station (BTS) or certain antenna of a BTS respectively. Thus, the received signal level become lower continuously until it falls underneath the minimum requirements for communication. Or error rate may grow due to interference, the distance to the BTS may be too high, all these effects may diminish the quality of the radio link and make radio transmission impossible in the near future.
- The wired infrastructure may decide that the traffic in one cell is too high and shift some MT to other cells with a lower load (if possible). Thus, handover may be due to load balancing.

Each cell area is covered by a transmitter, the cell radii may vary from tens of meters in buildings, and hundreds of meters in cities. The shape of cells is never perfect circles or hexagons, but depend on the environment (buildings, mountains, valleys, etc.) and sometimes on system load. The use of small size has several advantages: Higher capacity as it allows frequency reuse. If one transmitter is far away from one another, i.e. out side the interface range, it can reuse the same frequencies. The power needed in smaller cell size is less, which leads to less consumption of power in MT. The main disadvantage is an increase in handover rate as the cell size becomes smaller.

Much effort is being expended to study existing handover schemes, and to create new one that meet these challenges. If BTS has no ideal channels, it may drop handover request and cause forced termination of a call in progress.

1.1 ATM-BASED PCN

Future personal communication networks (PCN) will likely employ ATM-based backbone networks to interconnect cellular mobile networks. To support network wide handovers, new and handover call requests will compete for connection resources in both the mobile and backbone networks. In mobile networks, one common bandwidth resource access priority scheme is the guard channel scheme. Also access priority can be established via queuing scheme. For example, servicing handover calls and new calls with blocked calls queued and blocked calls dropped disciplines, respectively would enhance access priority for handover calls [14].

The fixed or dynamic guard channel methods can be extended to the nodes in the backbone network for link bandwidth allocation to enable prioritized handover calls, as well the hybrid scheme, i.e. use of guard channel with queuing method, would enhance the priority of handover calls.

1.2 MOBILE ATM [6][9]

In recent years, with the rapid development of wireless mobile communication, a new generation of wireless ATM mobile communication system has emerged. Its main differences from the non-ATM wireless access system are:

- It has an ATM network supporting terminal mobility, i.e. mobile ATM core network.
- Location and handover management that can support ATM and non-ATM mobile communication terminals.

The purpose of building mobile ATM core network is to provide a high-speed backbone communication network that can support mobile communication terminal location and handover management. Mobile ATM can also serve as a universal interface network for various mobile communication technologies. This is because ATM as a large traffic switching technique has relatively high performance-price ratio, and allows different types of traffic information to be transmitted in the same network. For example, in cellular mobile communication, voice information can be taken as ATM Adaptation Layer Type 1(AAL1), its packet information treated as constant bit rate (CBR),

IP(internet protocol) data as AAL5, its packet information connected as undefined bit rate (UBR) and available bit rate (ABR).

No matter which wireless access technique, mobile ATM can provide communication terminals with mobility support. This is because the lowest connection at ATM layer can be made at the wireless port, the received traffic information is processed using link layer mode in wireless channels, when the wireless port undertakes to transform the data format assigned by wireless link into a format adaptable to AAL. For example, in mobile ATM network, wireless port can provide access processing for WLAN (Wireless Local Area Networks), IP terminals can embed IP information in wireless ATM frame and send it to the wireless port, and the wireless port transfers the received information packet to ATM connection. Besides, IP terminal uses the mobility function of mobile ATM to map IP into ATM to complete handover and location management. Mobile ATM core network can support GSM, WLAN and WATM (Wireless ATM) access.

Mobile ATM architecture is composed of standard ATM network, wireless port, fixed or mobile communication terminals. The wireless port of the base station establishes communication links with mobile communication terminals through wireless channels. This network architecture allows ATM and non-ATM wireless addressing access.

1.3 PROBLEM STATEMENT

Handover initiated in PCN due to reasons mentioned above, a new channel has to be granted to handover request for successful handover. To keep forced termination probability to desired minimum values, handover algorithm should avoid blocking handover request due to lack of resources (radio and wired links). This could be achieved by giving handover high priority over initiating calls. This dissertation proposes a hybrid handover scheme, which aims to give handover calls higher priority than new calls. The hybrid scheme combines two priority schemes (queuing and channel

reservation schemes) and achieves a reduction in forced termination probability than what any scheme can give separately.

The network resources are limited due to physical limitation of wired link and frequency interference in radio link. Consequently, as forced termination probability decrease, the blocking probability of new calls increases. Careful implementation of handover algorithm leads to minimum forced termination probability and keeps blocking probability to the objective value.

1.4 ORGANIZATION OF DISSERTATION

Chapter –1, Gives an overview of the problem.

Chapter –2, Views various handover schemes.

Chapter –3, Describes the new proposed scheme.

Chapter –4, Describes the simulation models used in this dissertation.

Chapter –5, Results from simulation is obtained, and description of the results

Chapter –6, Gives a conclusion and scope for future work.

HANDOVER SCHEMES

Many handover schemes have been proposed in literature, in order to reduce handover failure and call forced termination. The reduction of handover failure, results an increase of new call blocking probability. It becomes a tradeoff between forced termination of ongoing call, and blocking a new originating call. However, the forced termination of ongoing calls is a less desirable event in the performance evaluation of a PCN network than blocking new calls. When a call is in progress in a cell, efforts are made to provide continuity to the current call when the user moves from one cell coverage area to another [12].

2.1 QUEUING HANDOVERS

Queuing of handover requests is made possible by the existence of the time interval the MT spends in the handover area, where it is physically capable of communicating with both the current and next BTSs. The fact that successful handover can take place anywhere during this interval marks a certain amount of tolerance in the delay for the actual channel assignment to the handover request [2]. During this interval the MT communication with the current BTS degrades at a rate depending on its velocity. The degradation is easily monitored by means of radio channel measurements, usually taken by the MT and submitted to the network [3]. The basic idea that the originating calls are not assigned channels until the queued handover requests are served.

2.1.1 FIFO Queuing scheme

In queuing handover, if all channels of a cell are occupied, calls originating within that cell are blocked and the handover requests to that cell are queued. FIFO is queuing discipline, in which, the call first queued, will be first served.

2.1.2 Measurement-Based Prioritized Scheme [3]

For successful handover, a channel must be granted to the handover request before the power received by the mobile terminal reaches the threshold, i.e., the threshold in the received power, below which acceptable communication with the BTS of the current cell is no longer possible. The handover area where the ratio of received power levels from the current and the target BTSs is between the handover and the receiver thresholds. If the power level of the current BTS falls below the receiver threshold prior to the MT begin assigned a channel by the target BTS, the call is terminated; therefore the handover attempt fails.

The queuing scheme previously suggested for protection of handovers utilize the FIFO queuing discipline, which does not consider the rate of degradation in the current radio channel. Measurement-based priority scheme (MBPS), is a queue discipline, which provides a significantly improved probability of successful handover by taking into account the degradation rate.

MBPS is non-preemptive dynamic priority discipline. The handover area can be viewed as regions marked by different ranges of values of the power ratio, corresponding to the priority levels such that the highest priority belongs to the MT whose power level is closest to the receiver threshold. On the other end, the MT that has just issued a handover request has the least priority. Obviously, the last comer joins the end of the queue. The power levels are monitored continuously, and priority of the MT dynamically changes depending purely on the power level it receives while waiting in the queue. A queued MT gains a higher priority as its power ratio decreases from the handover threshold to the receiver threshold. The MTs waiting for a channel in the handover queue are sorted continuously according to their priorities. When a channel is released, it is granted to the MT with the highest priority. A flow chart of MBPS is presented at Fig. 2.1.

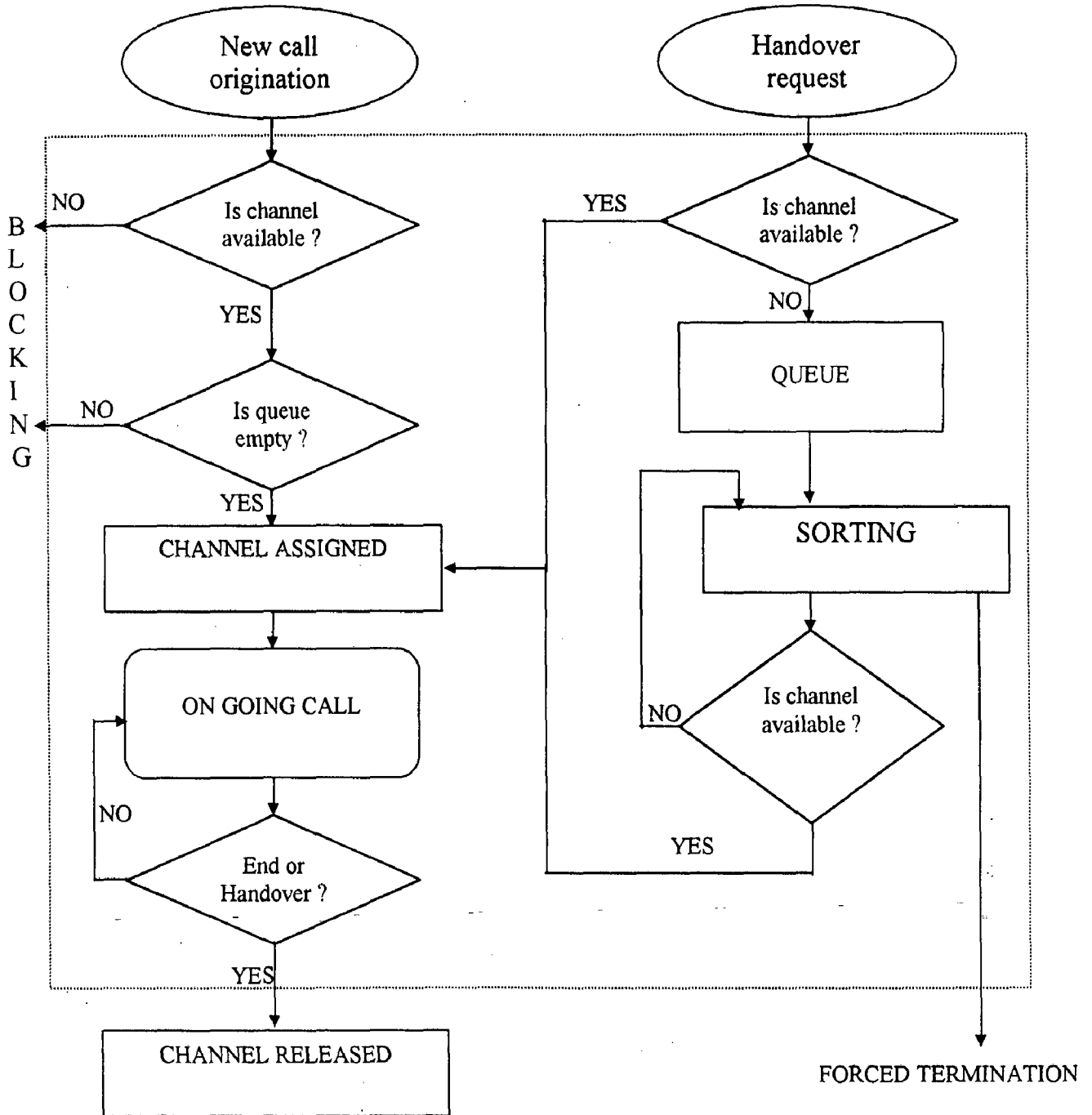


Fig. 2.1. Calls flow diagram for the measurement-based priority scheme

2.1.3 Signal Prediction Priority Queuing [11]

Signal prediction priority queuing (SPPQ), which uses both received signal strength (RSS) and the change in RSS (Δ RSS) to determine the priority ordering of an MT. In order to optimize the system for the minimum number of dropped handovers, the handover that would be terminated next should be the first to be handed over. Because this algorithm uses both RSS and Δ RSS (or speed and position) to determine the priority, it will perform better in a micro/picocellular environment than a pure RSS-based method (MBPS) or FIFO queuing.

In SPPQ, over every time interval Δt , the signal is averaged and stored for an additional Δt . The priority is determined by the intersection of the line created by extrapolating the two most recent average RSS measurements and finding its intersection with a “minimum allowable RSS” line (the RSS value for this line is f_{min}). The difference between the current time and the intersection time will be the “expected time for this call to be lost” and is denoted as t_L .

Any instantaneous value of t_L can be calculated using the following:

$$t_L = \frac{[f(t) - f_{min}] \Delta t}{[f(t - \Delta t) - f(t)]} \quad (2.1)$$

Where $f(t)$ is the RSS at time t , f_{min} is the minimum allowable RSS, Δt is the averaging interval (and the distance between sample points), and t_L is the expected amount of time before the call will be lost. Fig. 2.2 shows a flow chart of SPPQ.

2.2 BANDWIDTH RESERVATION BASED SCHEME

Handover calls can experience more favorable blocking probability than the new calls by bandwidth allocation during call admission using bandwidth reservation scheme, in which a part of the bandwidth is reserved exclusively to serve handover calls. This scheme is called guard channel scheme.

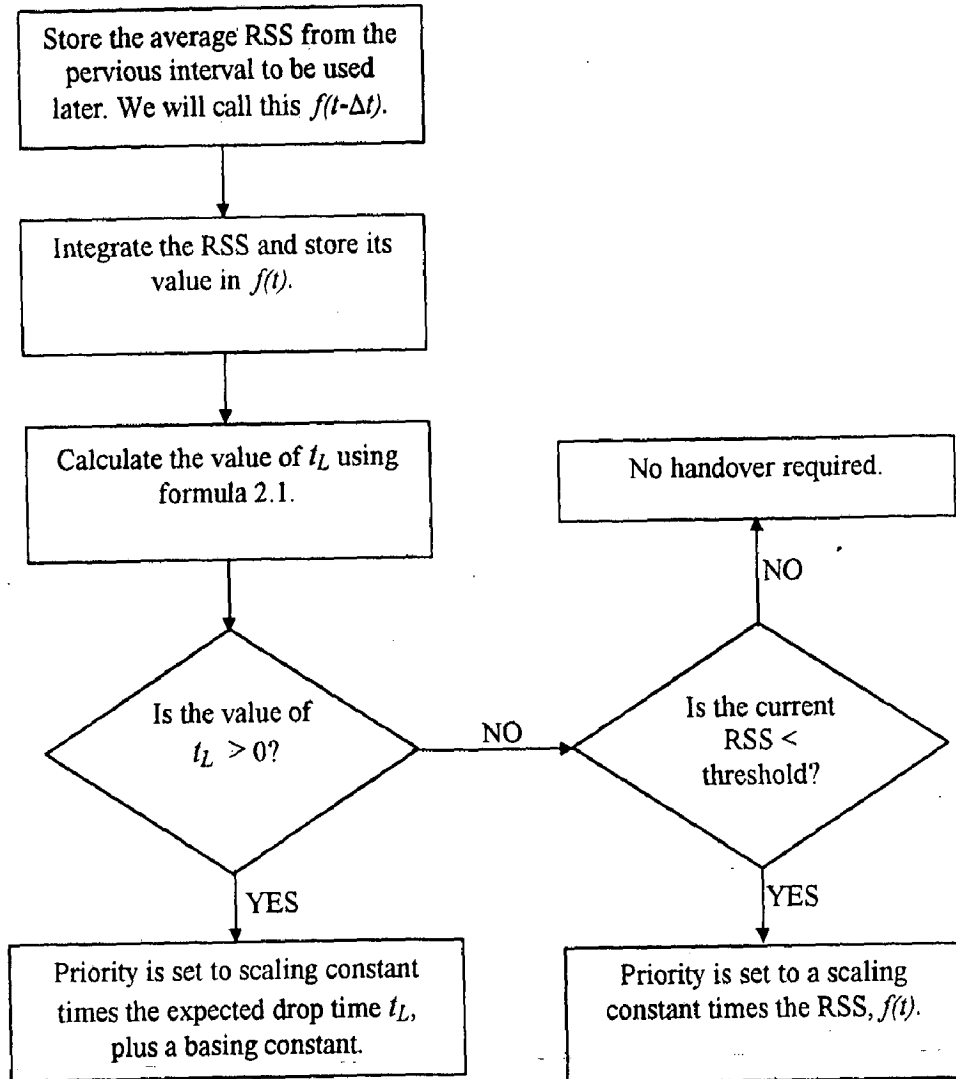


Fig. 2.2. Flow chart of SPPQ priority calculation. An extra case is added to prevent dropped calls by giving vehicles with very low RSS some handover priority even though their RSS is constant or increasing.

2.2.1 Fully Shared Scheme

The fully shared scheme (FSS) is employed by typical radio technologies that have been proposed for personal communications services (PCS). In FSS, the BTS handles the call requests without any discrimination between handover and new calls. All available channels in the BTS are shared by handover and new calls. Thus, it is able to minimize rejection of call requests and has the advantage of efficient utilization of wireless channels. However, it is difficult to guarantee the required dropping probability of handover calls, which is less desirable than restricting attempts of new calls for continuity of handover calls [2].

2.2.2 Reserved Channel Scheme

Reserved channel scheme (RCS) [2], gives handover calls a higher priority than new calls. In RCS, a number of wireless channels, called guard channels, are exclusively reserved for handover calls, and the remaining channels, called normal channels, can be shared equally between handover and new calls. Thus, whenever the channel occupancy exceeds a certain threshold, RCS rejects new calls until it goes below the threshold. Handover calls are accepted until the channel occupancy goes over the total number of channels in a cell. It offers a generic means to decrease the dropping probability of handover calls but causes reduction of total carried traffic. The reason total carried traffic is reduced is that fewer channels except the guard channels are granted to new calls. The use of guard channels requires careful determination of the optimum number of these channels, knowledge of the traffic pattern of the area, and estimation of the channel occupancy time distributions.

2.2.3 Dynamic Reserved Channel Scheme [2], [14]

The objectives of reserved channel scheme DRCS are to satisfy a desired dropping of the probability of handover calls, to reduce the blocking probability of new calls, and to improve the channel utilization. In DRCS, both handover and new calls share equally the normal channels, which are radio channels below the threshold. The

guard channels, the remaining channels above the threshold, are reserved preferentially for handover calls in order to provide their required QoS. Those channels, however, can also be allocated as much as the request probability for new calls instead of immediately blocking, unlike RCS. Thus, handover calls can use both normal and guard channels with probability of one if these channels are available. New calls use normal channels with probability of one, but guard channels can be used for new calls according to the request probability. It contributes to reducing the blocking probability of new calls and improving the total carried traffic. Fig. 2.3 shows the call processing flow diagram for DRCS.

The request probability reflects the possibility that the BTS permits new calls to allocate the wireless channel among the guard channels. It is dynamically determined by the probability generator in which the request probability is computed considering the mobility of calls, total number of channels in a cell, threshold between normal channels and guard channel, and current number of used channels. Among these factors, the mobility to calls is most important. The mobility of calls in a cell is defined as the ratio of the handover call arrival to the new call arrival rate [2].

Other DRCS are proposed in literature, the main idea in those schemes is when all normal channels are occupied, it allows the new call to use the reserved channels, according to certain criteria instead of blocking.

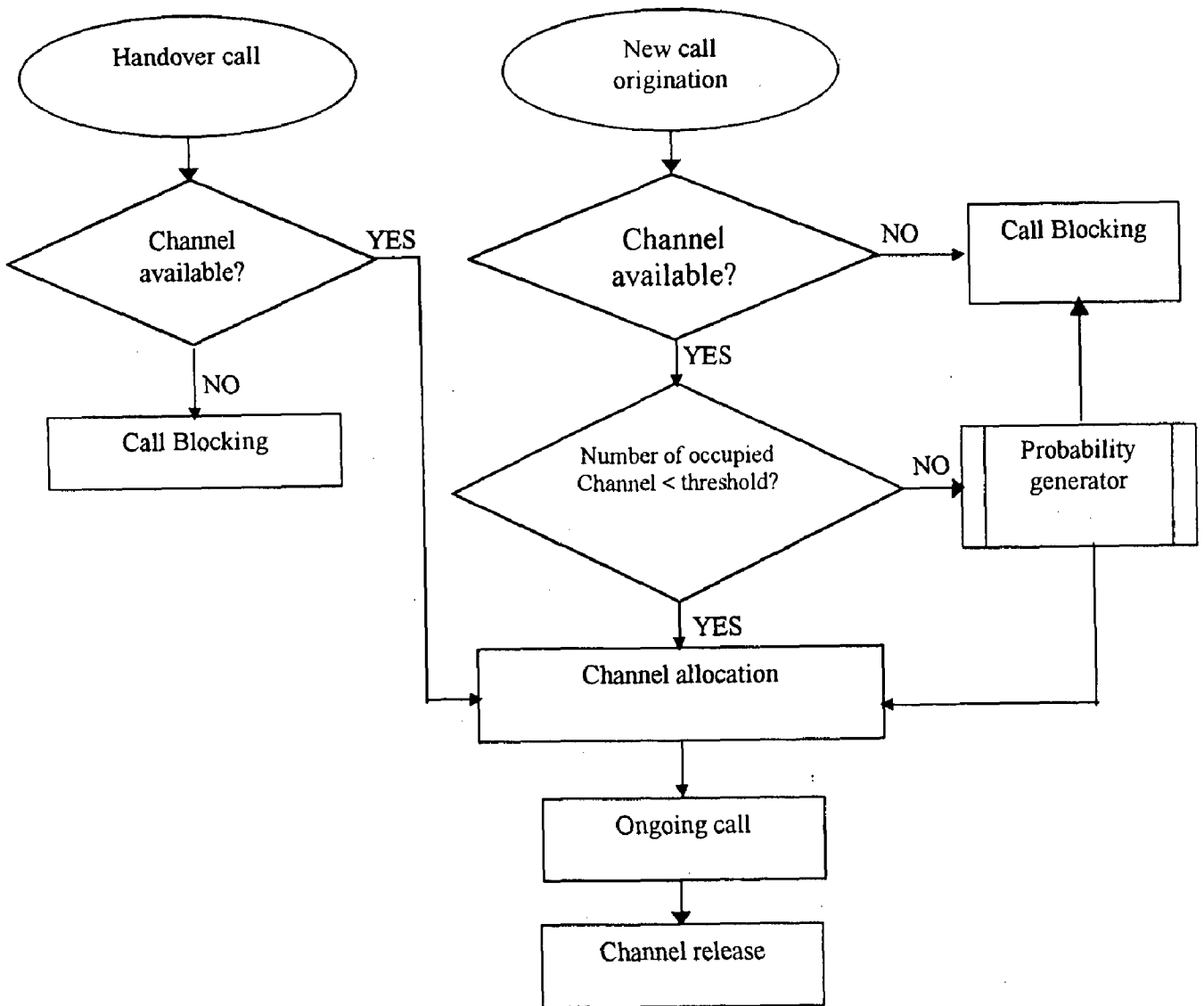


Fig. 2.3. A call processing diagram using DRCS.

2.3 THE SUBRATING SCHEME

In this scheme, certain channels are allowed to be temporarily divided into two channels at half the original rate to accommodate handover calls. This subrating occurs when all the channels occupied at the moment of handover arrival. When subrating channel is released, it forms into an original full-rate channel by combining with another subrated channel [13].

2.4 CHANNEL CARRYING AND BORROWING SCHEME

In channels borrowing scheme, if a handover request finds that all the channels are busy, it can borrow a free channel from its neighboring cells [4]. As users requesting handover always occupy a channel in its current cell. then the channel could be carried into new cell [5], so handover would not be blocked. When it is said that a channel is carried into a new call, this means that the mobile user continues to use this channel, but now communicates with the base station in the new cell.

A HYBRID HANDOVER SCHEME

The hybrid handover scheme, proposed in this dissertation, combines guard channel scheme (bandwidth reservation) and handover queuing scheme. In this work, FIFO and MBPS queuing discipline [3][11], and RCS [2] is used. However, DRCS could be used for better network utilization, and reducing the new calls blocking probability.

3.1 ADMISSION CONTROL

In general, given total resources (channel or bandwidth) that may be allocated to the new and handover calls, blocking occurs during call admission control, when a call requires bandwidth over the radio channel in a cell or the link traversed over the backbone network in excess of what is available. Without prioritized allocation scheme, handover and new calls would have the same blocking probability [14].

In this scheme, a new call is admitted only if number of free channels is more than number of guard channels, otherwise, the new call is blocked. Handover calls are admitted if any channel is free. If all the channels are occupied, then the handover is queued using queuing discipline like FIFO and MBPS queuing schemes. Handover requests are blocked only if it is waiting in the queue for free resource, and the tolerance time period elapsed before granting a free resource. This reflects the natural boundary of the queue size.

Fig. 3.1 shows a call flow diagram of hybrid of queuing scheme and RCS. Queuing scheme gives the priority to handover calls by keeping them waiting for resources to be freed, and give them priority over the new calls, while, RCS gives priority to the handover calls by preventing new call to use certain number of channels, which are reserved exclusively for handover calls. The hybrid scheme, combines the priority from the both schemes, and gains a higher priority for handover calls.

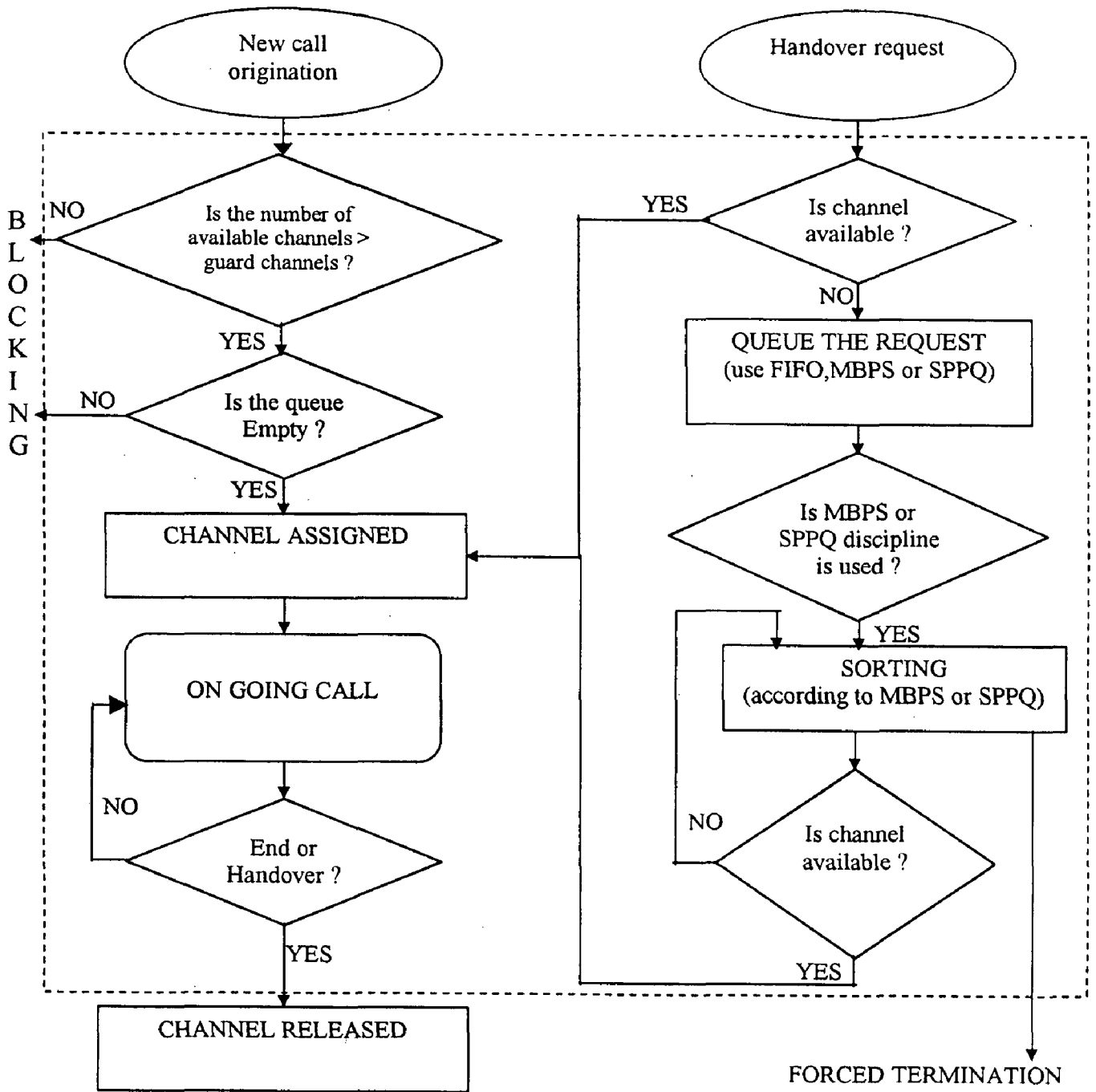


Fig. 3.1. Calls flow diagram for the hybrid of queuing scheme with the guard channel scheme.

3.2 EXTENSION OVER ATM-BASED NETWORK

The fixed or dynamic guard channel method can be extended to the nodes in the backbone network for link bandwidth allocation to enable prioritized handover admission control. Different bandwidth assignment schemes may be employed depending on the traffic characterization and multiplexing scheme. Cellular mobile systems employ channel assignment, whereas ATM-based backbone networks may employ statistical multiplexing and statistical bandwidth assignment. Since statistical bandwidth assignment can be mapped into per call equivalent bandwidth assignment, the concept of guard channel, i.e. RCS can be directly applied to set aside reserved bandwidth for handover calls [14].

3.2.1 Admission Control Over backbone Network

New calls may need to use the backbone network to communicate with other call parties, served by different mobile ATM switch (MAS). A new call is admitted only if radio and backbone resources are available, otherwise, the new call is blocked. Fig.3.2 shows flow diagram for call admission control over radio and backbone links. The handover calls, may also need to use a backbone link, in this case, RCS may be applied for both radio links and backbone links, as stated by [14].

Each cell is served by BTS, which has N_R radio channels to serve MT in the cell. Number N_{RG} of radio channels, can be reserved to serve handover requests. On the other hand, each backbone link has N_L channels, number in channels in a backbone link, is relatively larger than the number of channels in a BTS. As at the radio channels, N_{LG} guard channels can be reserved to serve handover calls, that request backbone link. The number of guard channels should be determined carefully in both radio and backbone links; this number depends on the traffic patterns [3] and network topology. DRCS may be used in both radio and backbone link to provide efficient utilization of network resources.

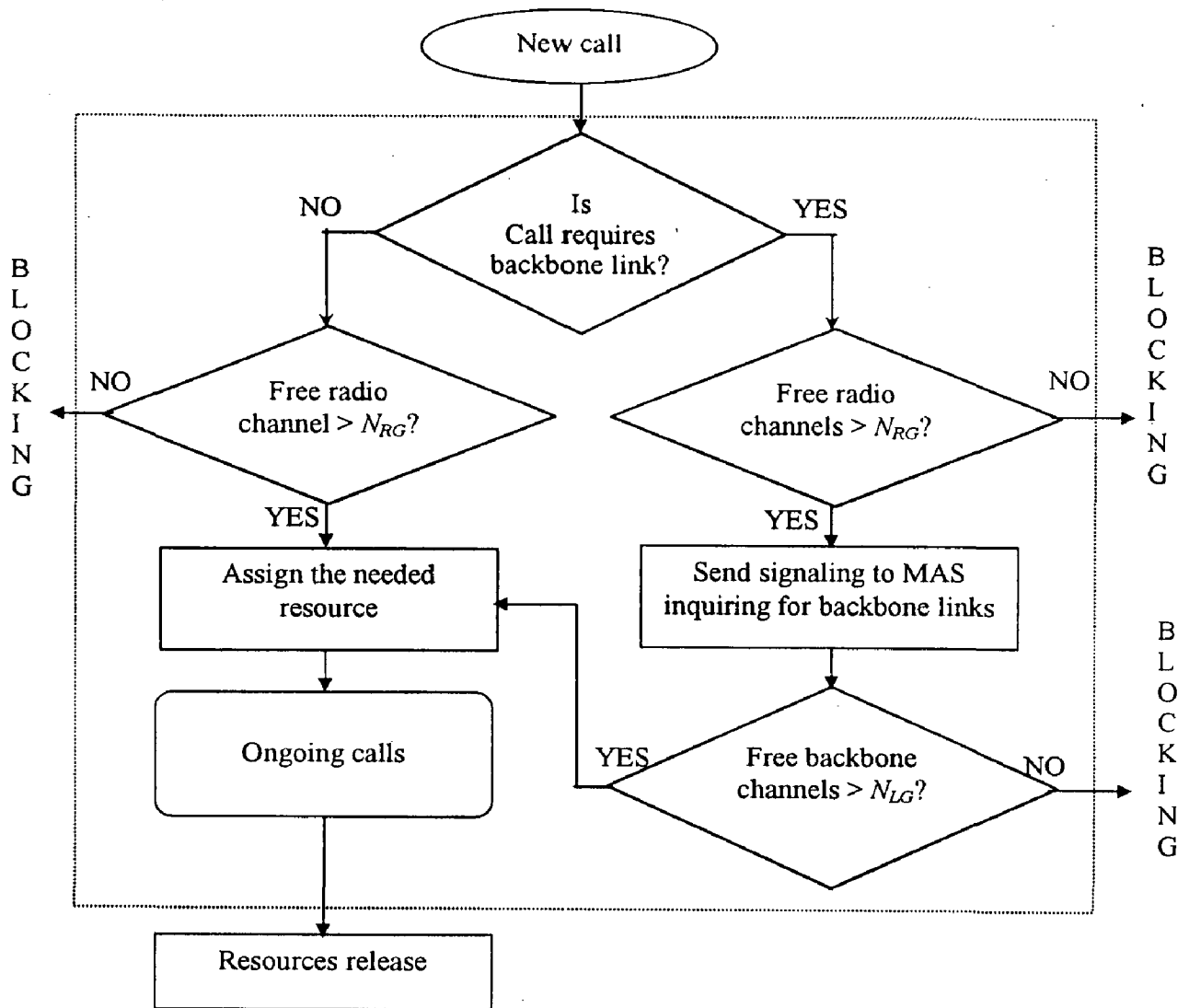
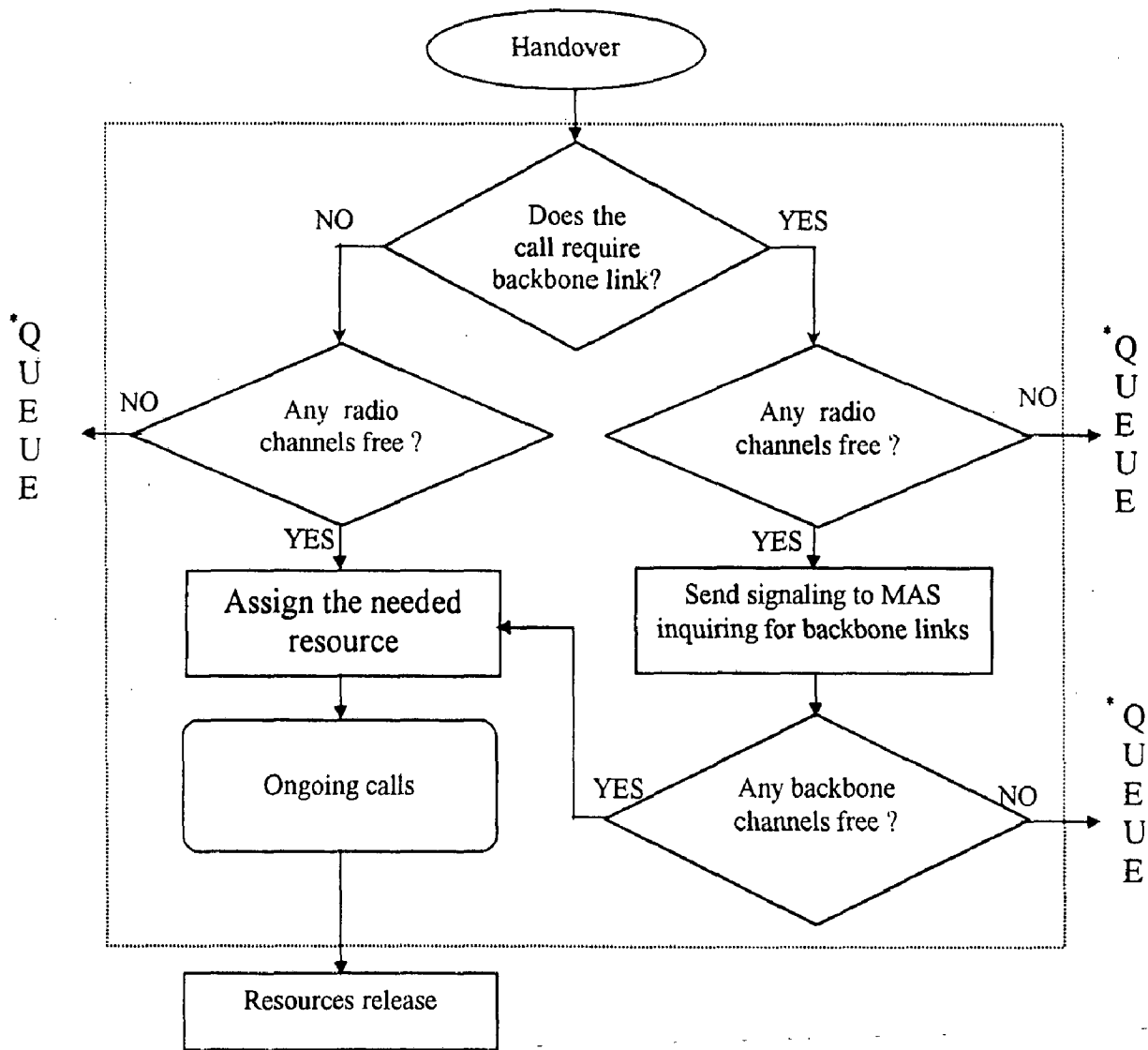


Fig. 3.2. New calls admission control flow diagram for RCS considering both radio and backbone links.



*. Queuing the handover request will be according to queue scheme as shown in Fig. 3.1.

Fig. 3.3. Handovers admission control flow diagram for RCS considering both radio and backbone links

Queuing the handover request, which is used in considering radio link only, can be used in the extension to ATM-based network. The idea is that when resources are needed to serve handover are not available, is to queue handover request instead of blocking the handover call, the queuing is limited by a time interval, during which some resources are expected to be freed, so handover request can be served. If handover request needs a backbone link, then it is queued if radio resources are not available or backbone link resources are not available. Fig.3.3 shows flow diagram for handover admission control over radio and backbone links.

3.2.2 Backbone Handover

The physical network underlying the assumed connection architecture is shown in Fig. 3.4. The radio access points (RAP) providing the physical radio coverage each are connected to a local MAS, thus forming one elementary mobility zone per MAS. Wide area mobility and consequently inter-zone handover and fixed network interworking is facilitated through the concept of an cross over switch (COS). While the MAS takes care of intra-zone handover, the COS is responsible for processing inter-zone handover requests [10].

A call may need one or more backbone link, to communicate with its destination. The number of links depends on the network topology and the destination of the call. The cost of a call is associated with the number of links, so efforts are made to optimize the number of links to the minimum. A partition methodology is proposed in [15], in which the set of cells are divided into subsets so that the base stations of the cells belonging to the same subset are connected to the same ATM switch. The optimum partition, which gives the minimum number of handover requires a change in the ATM switch. This methodology is based on handover rate between cells, which allows the choice of the best partition by taking mobility, radio, and fixed network aspect into account.

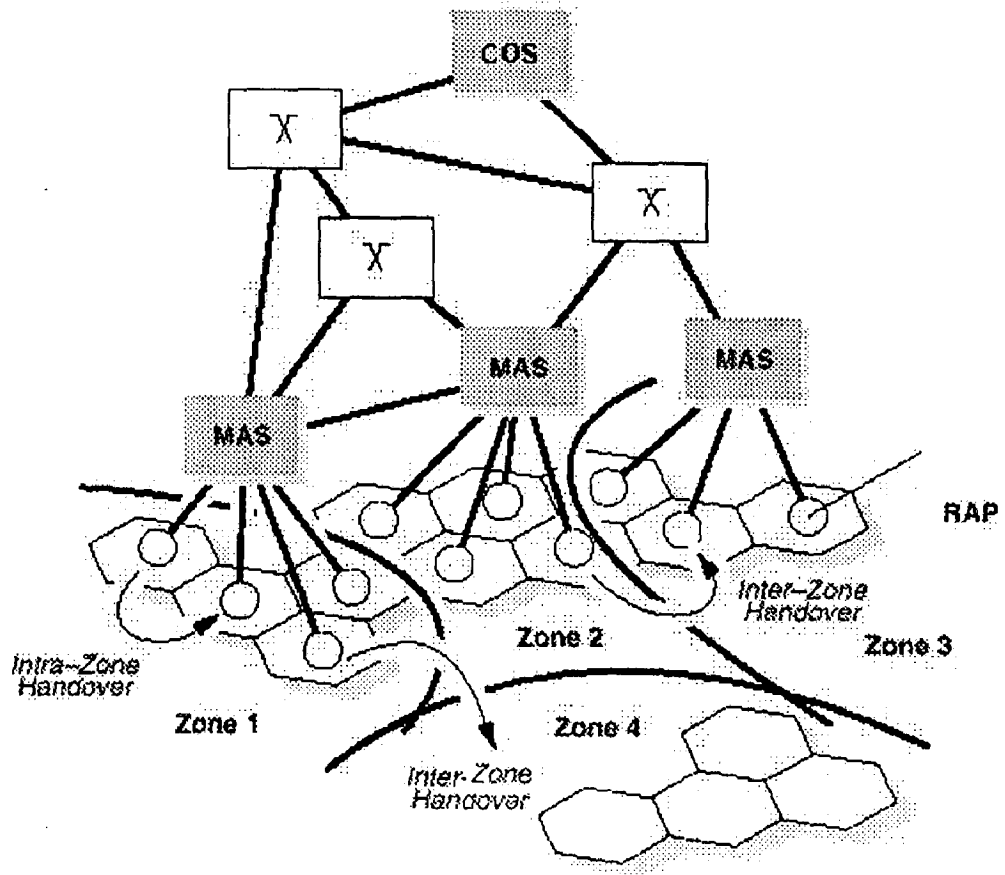


Fig. 3.4. Handover requires backbone link.

SIMULATION MODEL

In this dissertation, two simulation models are proposed. The first one simulates the environment of PCN, using single cell, and without considering the connection to the backbone network. Simulation of a call handling operation within one single cell is sufficient to generalize the results to complete service area of mobile switching center (MSC). Because the traffic load is allowed to vary, single-cell model can realistically reflect the behavior of a real cellular system [3][11]. The second model considers multiple cells connected by MAS, which are connected by fixed backbone network.

Both simulation models can work with any channel assignment strategy. However, the results are obtained using fixed channel assignment (FCA).

4.1 TRAFFIC MODEL

Traffic in a cell consists of new calls initiated inside the cell and handovers arriving to the cell from the neighboring cells. New calls and handovers follow Poisson distribution. The offered load (i.e. traffic) is variable, to obtain different points, while the fraction of total traffic due to the handover is kept fixed. Call duration is assumed to be exponential.

When a new call is originated in a cell and assigned a channel, the call holds until it is completed in the cell or handed over to another cell as the mobile moves out of the cell, Fig.4.1 shows traffic flow into a cell. MT stays in the coverage area of a cell for a period of time (dwell time) that is exponentially distributed, and then it moves to one of surrounding cells.

The probability of requiring a handover depends on the cell coverage area, the MT movement, and the call duration. A call handover must be directed to one of the neighboring cells. The probability of each neighboring cell receiving the call depends on

the amount of common boundary area and mobile direction [14]. In both simulation models, we consider typical hexagonal cell, and we assume that the neighboring cells receive the handover with an equal probability of 1/6 for each.

4.2 SIMULATION MODEL I

In this model, a single cell is considered, the arrival of new calls and handovers is as mentioned above. Fig.4.1 shows the traffic flow into the test cell.

4.2.1 Simulation Parameters

The simulation parameters used are as follows:

N_R : Number of radio channels in each cell.

N_{RG} : Number of radio guard channels in each cell.

λ_o : New call arrival rate.

λ_{hi} : Handover call arrival rate.

ρ : The offered load which is $\lambda_o + \lambda_{hi}$.

t_c : New call holding time.

t_h : Handover call holding time.

t_q : Maximum tolerable time in the queue.

4.3 SIMULATION MODEL II

In this model, model I is extended to consider ATM-based backbone network, which connects MAS. PCN architecture based ATM switches proposed in [7], as shown in Fig. 4.2. ATM technology is suited to the infrastructure to interconnect the BTS of the PCN.

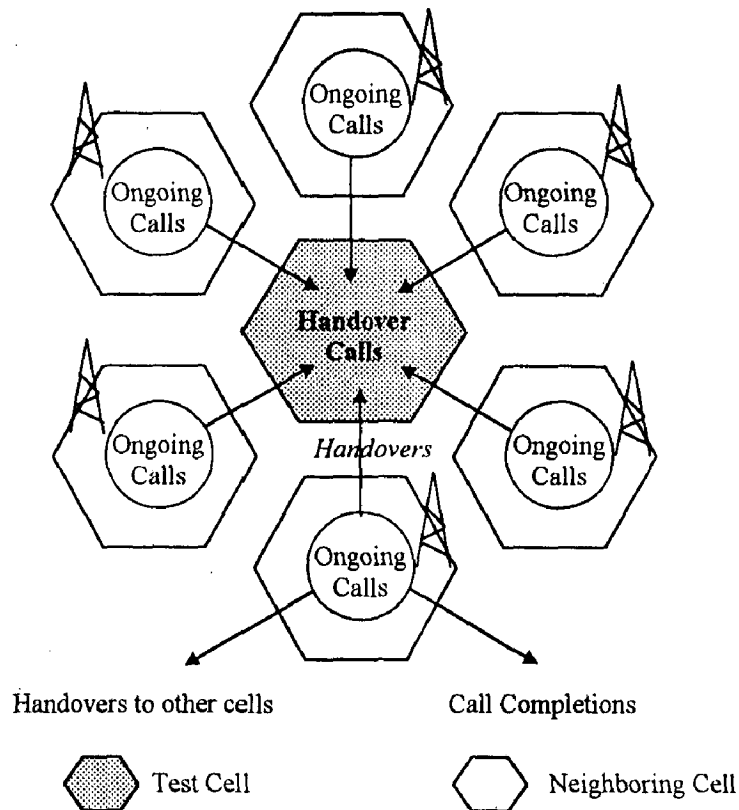


Fig. 4.1. Simulation model for single-cell

4.3.1 Environment Description

Each microcell has a BTS to serve the MT within the cell. The geographical area is partitioned into a set $C = \{C_1, C_2, \dots, C_n\}$ of n disjoint clusters, each cluster consists of a set of microcells. An ATM switch is allocated within each cluster and each BTS in this cluster is connected to the port of this switch. The ATM switch offers the service of establishing / releasing channels for mobile terminal in the cluster, also this switch should have routing/rerouting capabilities. Two neighboring clusters can be interconnected via the associated ATM switches. The links between ATM switches are called backbone links, and the links between ATM switch and base station are called local links.

An ATM-based topology could be represented by an undirected graph $H = (V, F)$; where each vertex v_i in V stand for a cluster C_i (or an ATM switch) and an edge $e_{i,j}$ is in F if clusters C_i and C_i are adjacent in the given network. Fig. 4.2 shows an ATM based PCN topology, which consist of 21 cells, attached to 8 ATM switches, which connected by 9 backbone links. In [7], they have given PCN with different number of cells and ATM switches configuration. In our simulation program, any topology could be adopted, with any number of cells, ATM switches, and backbone link. Corresponding graph of Fig. 4.2(a) is shown in Fig. 4.2 (b).

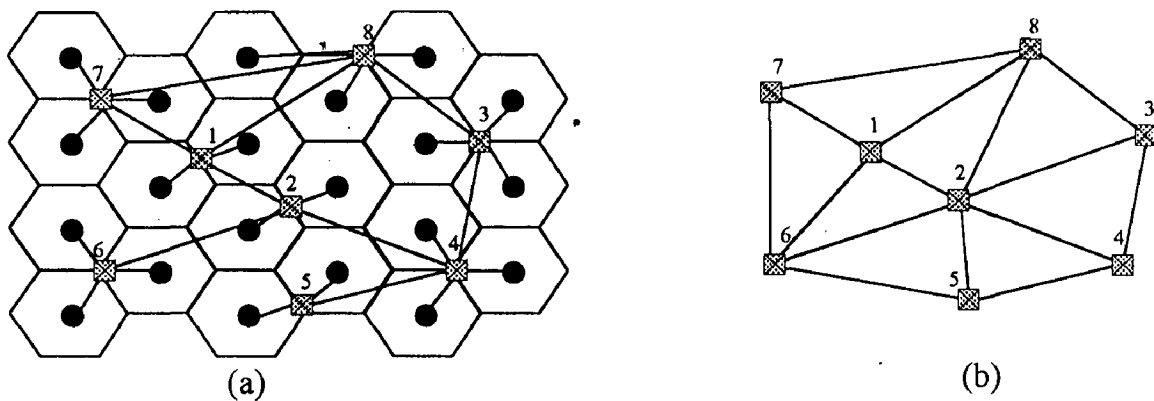


Fig.4.2. An ATM based cellular network. (a) An ATM based architecture. (b) Corresponding H graph of the PCN architecture.

Constructing a backbone network between MASs could be done in different ways. Depending on the geographical area, the cost of the backbone link, and traffic patterns. Fig 4.3 shows different possible backbone network for graph 4.2(b).

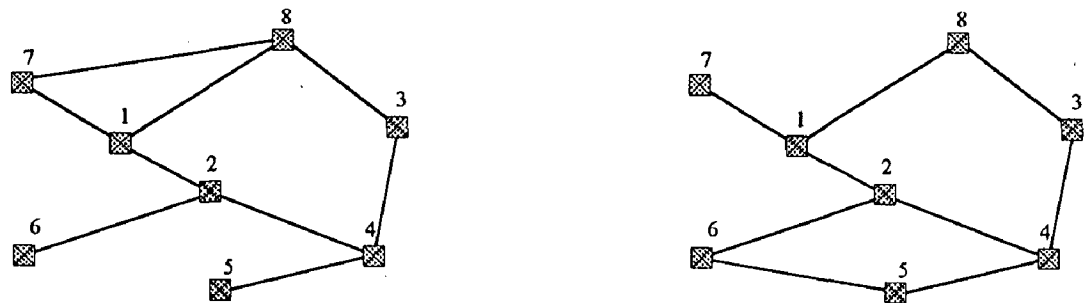


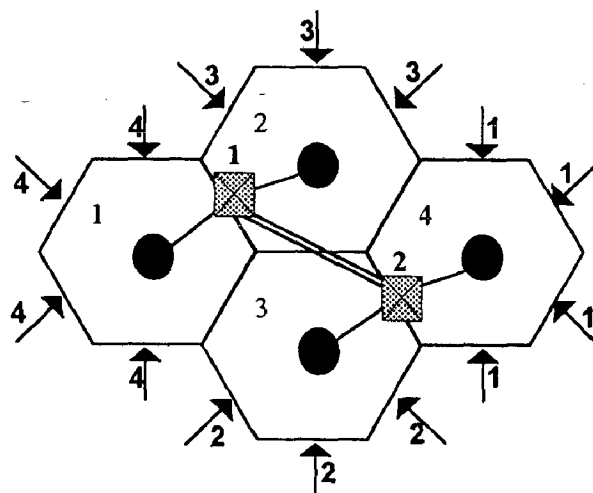
Fig. 4.3. Different possible backbone network connections.

MT engaged in a call or data transfer within the same cluster will consume two local links, one for each local link between base station and the associated switch. For intercluster communication, backbone links will be allocated in addition to local links. The channel occupied will depend on the communication path being assigned [7].

4.3.2 Simulation Environment

In the simulation, we will simulate the traffic in four cells as a part of full network. From fig. 4.2 we consider the ATM switches 1 and 2. BTS 1 and BTS 2 form a cluster, and connected to ATM switch 1, BTS 2 and BTS 3 form a cluster and connected to ATM switch 2. ATM switches 1 and 2 are connected by backbone link; this configuration is illustrated at Fig. 4.4. To eliminate the boundary effect, wraparound topology is used [12].

Traffic in the backbone link is from: calls between (BTS1 or BTS2) and (BTS3 or BTS3), and vice versa. And load from other parts of the network that may use this link in its communication. The number of channels available in this backbone network is relatively larger than that of each BTS radio channels. The initiated and handover calls in the cells, have Poisson rate as described above.



*Arrows symbolize handovers from surrounding cells


 Mobile ATM switch

Fig. 4.4. The simulated environment ATM-based cellular PCN

4.3.3 Mobility Model

MT may handover to cell which is inside or outside the four cells under simulation, this makes a radio channel release in that BTS, and may or may not effect the load in the backbone link; depending on the network topology and the destination. In our work we'll consider this has no effect on the backbone link.

We assume that the call handover from cell 1 to the neighboring cells, to cell 2 and cell 3 with probability $1/6$ for each one, the remaining four sides can be wraparound to cell 4 with probability $4/6$. And for cell 3, handover to cell 1, cell 2, and cell 4 with probability of $1/6$ for each one, the remaining three neighboring sides can warp to cell 2 with probability of $3/6$, so the total handover probability from cell 3 to cell 2 is $4/6$. Cell 4 is treated as the case of cell 1, and cell 2 as the case of cell 3. Fig.4.4 illustrates this wraparound topology.

Each cell has a BTS, which acts as a RAP, between MT and the core ATM network. BTS has a limited number of radio channels. Local link has a number of wired channels, which is equal to radio channels. Backbone link has a limited number of wired channels. For simplicity, the roaming of mobile terminal in the test cells only is considered.

The destination of the call is important to determine the need for local and backbone links, we define three call types according its destination, with probability of occurrence in the simulated environment, as follows:

- In cell call: in which the call source and destination at the same cell, the probability of this call is $P_{cell}=1/7$, MT consumes only radio channel.
- In cluster call: in which the call source and destination at the same cluster, and different cells, probability of this call is $P_{cluster}=1/7$, MT consumes radio channel and local link channel.
- Out cluster call: in which the call sources and destination at different clusters, probability of this call is $P_{backbone}=5/7$, MT consumes radio channel, local link channel and backbone link channel.

Note that, there is no competition for local links, if the radio channel is available, then local link is granted; because only the user in the cell may use the local link channel.

The call will occupy resources according to the probability above. If there are not enough resources available then, the call will be blocked. The new call may handover to the neighboring cell releasing the radio channel and local link channel, but may or may not release the backbone link depending on the destination and rerouting algorithm used.

4.3.4 Simulation Parameters

Following simulation parameters are used, in addition to the parameters which are used in the model I:

N_L : Number of backbone channels in each backbone link.

N_{LG} : Number of backbone guard channels in each backbone link.

P_{cell} : Probability of in cell call.

$P_{cluster}$: Probability of in cluster call.

$P_{backbone}$: Probability of out cluster call.

4.4 SIMULATION PROGRAM

The simulation program is implemented in Turbo C++ language, version 3.0, and run under MS DOS 6.2 environment on Intel Celeron personal computer. The object oriented approach is used to implement the real object. Simulation is discrete event simulation, as described in [8]. Results are directed to text file and graph obtained using MS EXCEL 2000.

RESULTS AND DISCUSSION

In this chapter, simulation results are obtained to evaluate the proposed hybrid scheme. Simulation program run for model I and model II using default values of simulation parameter to obtain the results.

The default values for the simulation parameter are defined as follows [11]:

$N_R= 30$ Radio channels in each cell.

$N_{RG}= 3$ Reserved radio channels in each cell.

$t_c= 60$ seconds average of new call holding time.

$t_h= 30$ seconds average of handover call holding time.

$t_q= 10$ seconds average time in the handover queue.

Handover has 50 % of the total traffic.

5.1 BLOCKING IN FSS

In FSS (or non- prioritized) new calls and handover are treated equally without assigning any priority to handover calls. Simulation shows that non-prioritized scheme handover and new calls have the same blocking probability.

As the network resources are shared equally between the handover and new call request and the handover has 50 % of the total traffic in the system. Handovers and new calls have the same blocking probability as shown at Fig. 5.1.

5.2 RESULTS IN SIMULATION MODEL I

The simulation model I was run with the default simulation parameter values. 100000 calls were sampled in the test cell.

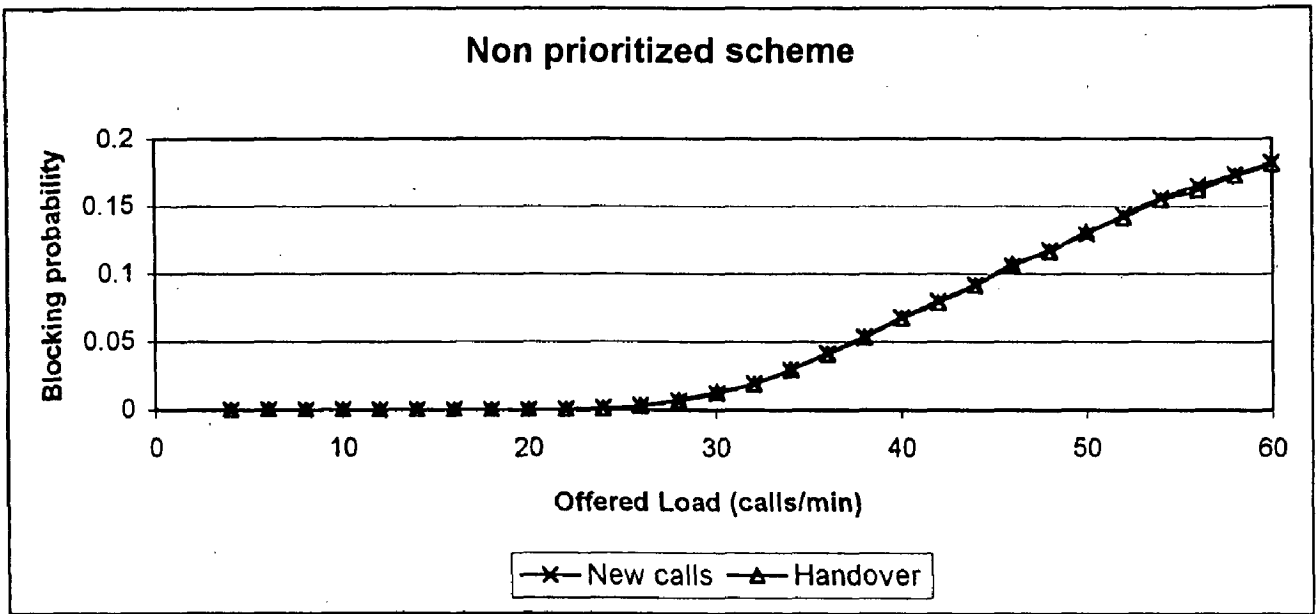


Fig. 5.1. Blocking probabilities for new calls and handovers of non-prioritized scheme, using default parameters values.

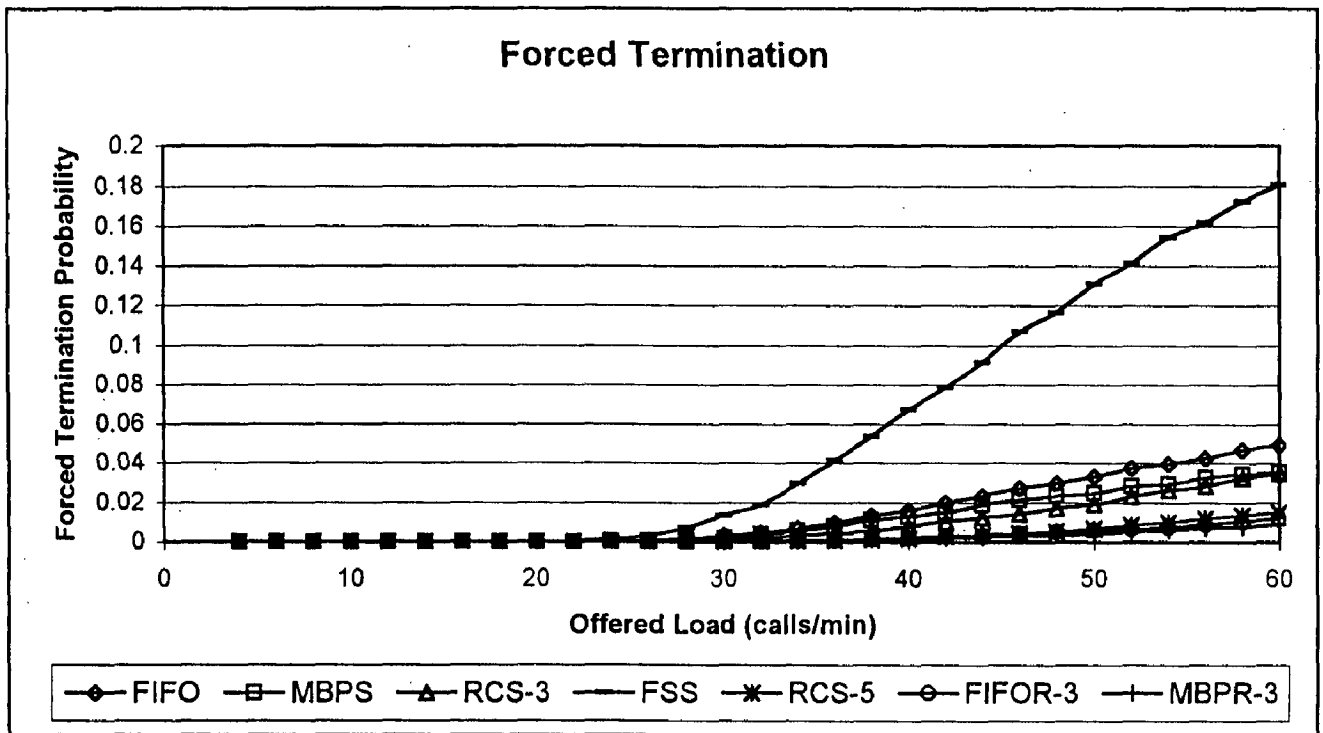


Fig. 5.2. Compare of forced termination probability for various handover schemes. Using the default parameter values.

5.2.1 Forced Termination Probability

The forced termination probability is the probability that an ongoing call be lost due to handover failure. Less forced termination probability is desirable for user satisfaction.

In Fig. 5.2, it is noticed that the highest forced termination probability is for the FSS. The values of forced termination probability in descending order with respect to the respective scheme are: FIFO, MBPS, RCS with 3 channels reserved for handover (RCS-3), RCS with 5 reserved channels (RCS-5), hybrid of FIFO and RCS-3 (FIFOR-3), and hybrid of MBPS and RCS-3 (MBPR-3).

The offered load varies from 4 call/min to 60 calls /min, which is considered as an overload traffic to the system. MBPR-3 scheme has the value of 0.0086 as forced termination probability at this load, where forced termination probability for MBPS is 0.0354.

The schemes competing for the better performance are MBPR-3, FIFOR-3 and RCS-5. Fig. 5.3 gives a closer look the behavior the three schemes.

Fig. 5.3, shows that the hybrid schemes have the least forced termination probability, despite that there is 5 channels reserved in RCS, and only 3 channels reserved in the hybrid schemes. When the load is low, all the schemes have approximately same behavior. MBPR-3 has the best performance over all other schemes. RCS-5 and FIFOR-3 show the same performance for low and moderate load, FIFOR-3 shows improvement over RCS-5, as the load increases, at load 50 calls/min FIFOR-3 starts showing noticeable improvement over the RCS-5.

5.2.2 Blocking Probability

The blocking probability is the probability that the new call finds all the channels busy, and blocked. Its important to keep track of the blocking probability, to see how much various schemes yield blocking probabilities. Increase in blocking probability is always the price we have to pay for decrease of forced termination probability. It is a tradeoff between the handover forced termination and new call blocking.

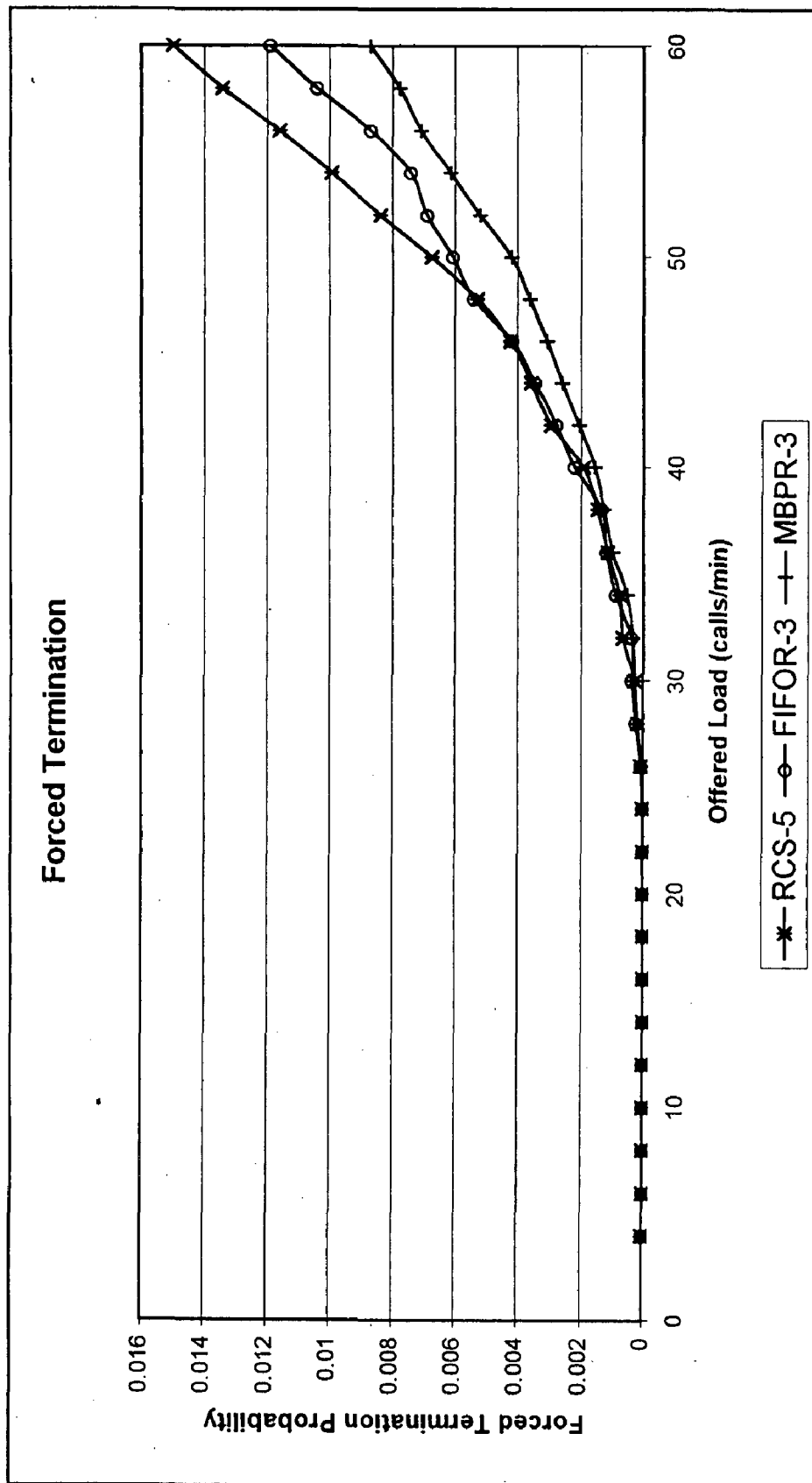


Fig. 5.3. Forced termination probability of MBPR-3, FIFOR-3 and RCS-5. Using the default parameter values.

Fig. 5.4 shows that RCS-5 has the highest blocking probability, over all other schemes. In Fig. 5.3 we have seen that the forced termination probability of RCS-5 is more than the forced termination of hybrid schemes (FIFOR-3 and MBPR-3). In this context, the proposed scheme achieves a significant improvement over the RCS in both less forced termination probability and less new calls blocking probability.

FSS scheme has the least blocking probability, because this scheme divides the resources equally between the handover and new calls. MBPS and FIFO schemes have approximately the same blocking probability. Fig. 5.3 shows MBPS has less forced termination probability compared to FIFO scheme.

RCS-3, FIFOR-3 and MBPR-3 schemes approximately have the same blocking probabilities. However, RCS-3 shows slightly less blocking probability than the hybrid schemes (FIFOR-3 and MBPR-3). On the other hand, the hybrid scheme has significant improvement over the RCS-3, as shown by Fig. 5.2.

5.2.3 Improvement

The improvement study carried out to show how much improvement is achieved by using the hybrid scheme in the comparison to the other schemes. The improvement reflects the reduced percentage of forced termination probability due to handover failure. The improvement of scheme S_1 over scheme S_2 is calculated as follows:

$$improvement (S_1, S_2) = \frac{(f(S_2) - f(S_1))}{S_2} * 100 \% \quad (5.1)$$

Where $f(S)$ is the forced termination probability by using scheme S .

Substitute S_1, S_2 for various schemes.

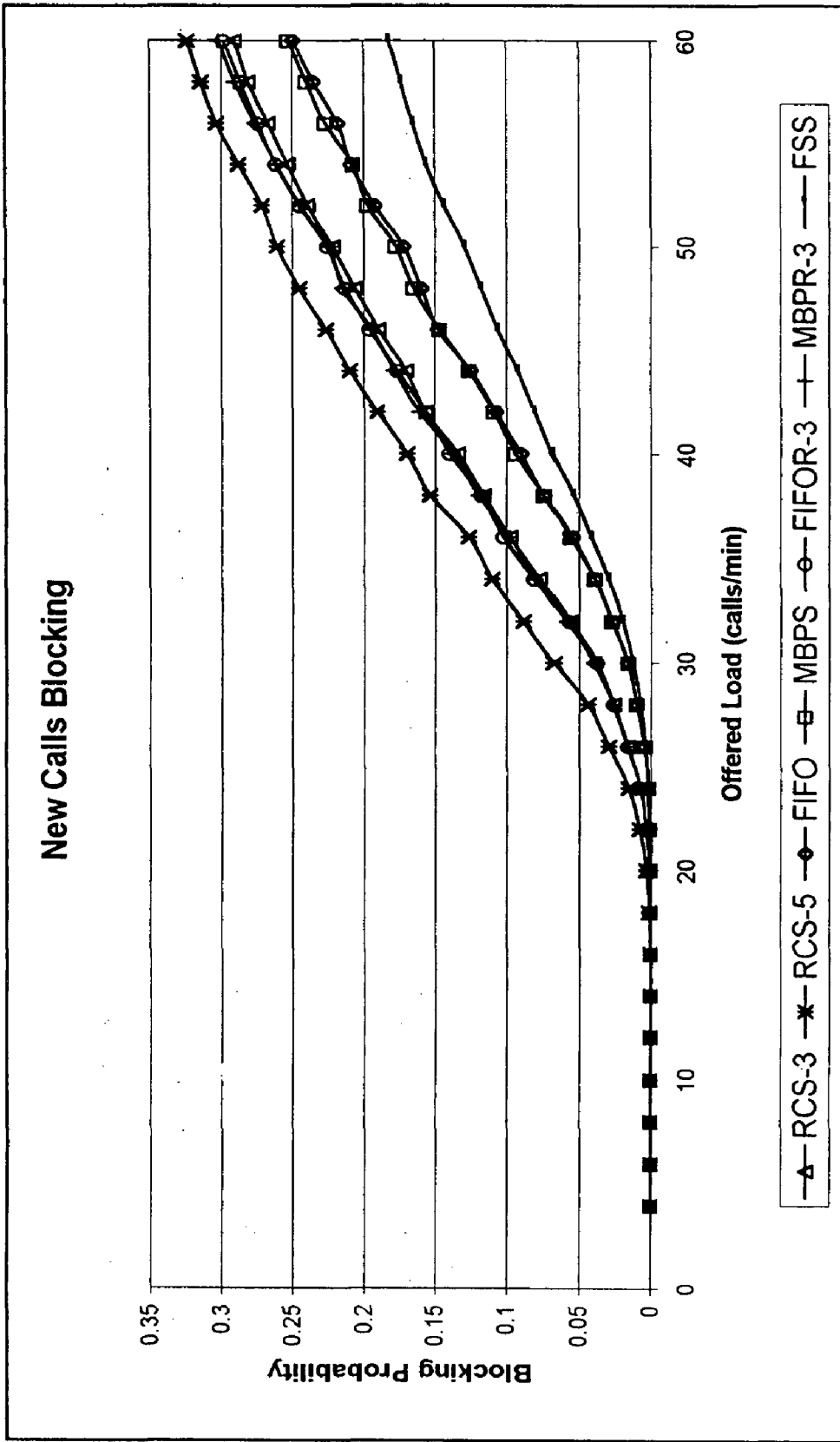


Fig. 5.4. Comparison of new calls blocking probability for various handover schemes. Using the default parameter values.

Fig. 5.5 shows improvement percentage of hybrid scheme compared to other schemes. The maximum improvement of MBPR-3 scheme over MBPS is 96 %, and average of 59 % of improvement. The maximum improvement of FIFOR-3 scheme over FIFO is 96.7 % and average of 58.9 % of improvement. The maximum improvement of FIFOR-3 scheme over MBPS is 96 % and average of 56 % of improvement

Fig. 5.6 shows improvement of MBPR-3 scheme over other schemes. The maximum improvement achieved by this hybrid scheme over RCS-5 is 66.7 % and average improvement is 19.7 %. The graph oscillated between high and low improvement values at moderate load; this means that the RCS-5 has some forced termination probability values near to values of MBPR-3 scheme, and other values much less than MBPR-3 scheme values. MBPR-3 scheme has improvement over FIFOR-3 scheme because the MBPS has less forced termination than FIFO scheme. The maximum improvement of MBPR-3 scheme over FIFOR-3 scheme is 41.4 % and the average improvement is 14.3 %. The maximum improvement of MBPR-3 scheme over RCS-3 is 88.9 % and average improvement is 54.3 %.

5.2.4 Blocking Increase

As consequence of reduction of forced termination probability, an increase in new calls blocking probability is introduced. The number of resources is limited (i.e. radio channels) as more channels are assigned to serve handover request, blocking probability will increase. We have studied how much the hybrid scheme introduces increase in blocking probability, in comparison to other schemes. The increased blocking reflects the percentage increase in blocking probability due to non-availability of resources of scheme S_1 over scheme S_2 , the increase is calculated as follows:

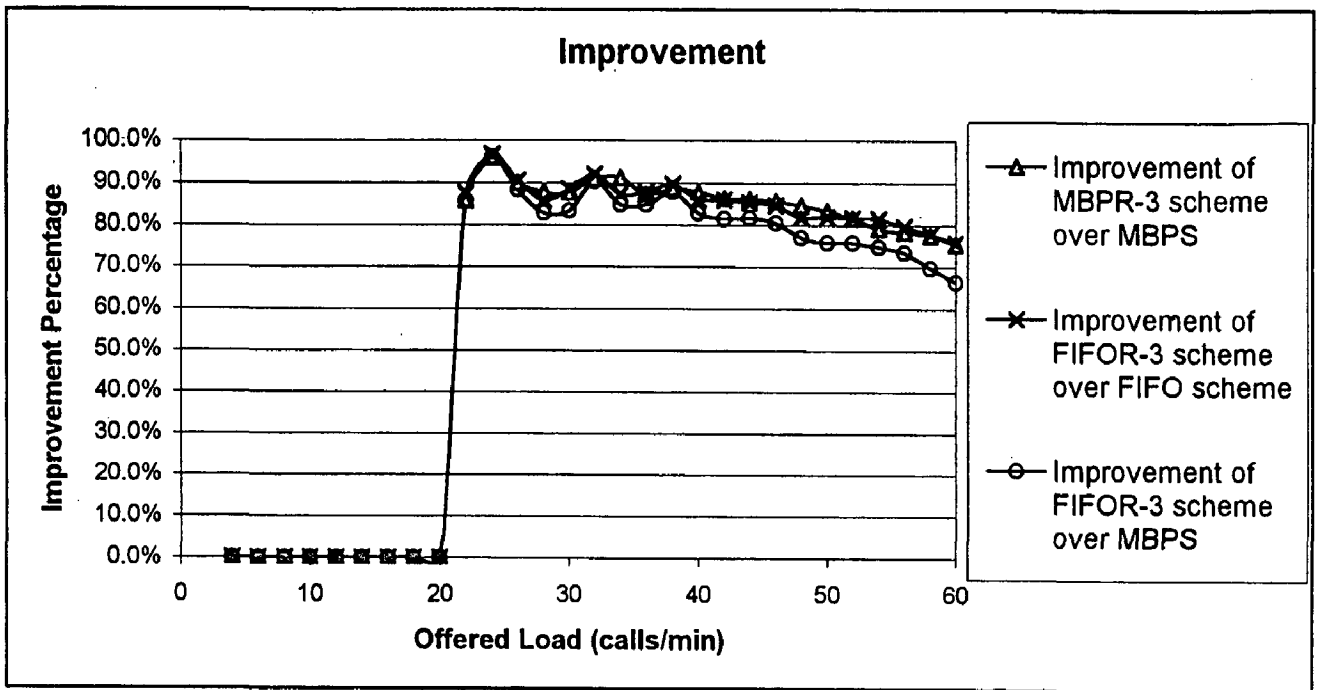


Fig. 5.5. Improvement of hybrid scheme over other schemes.

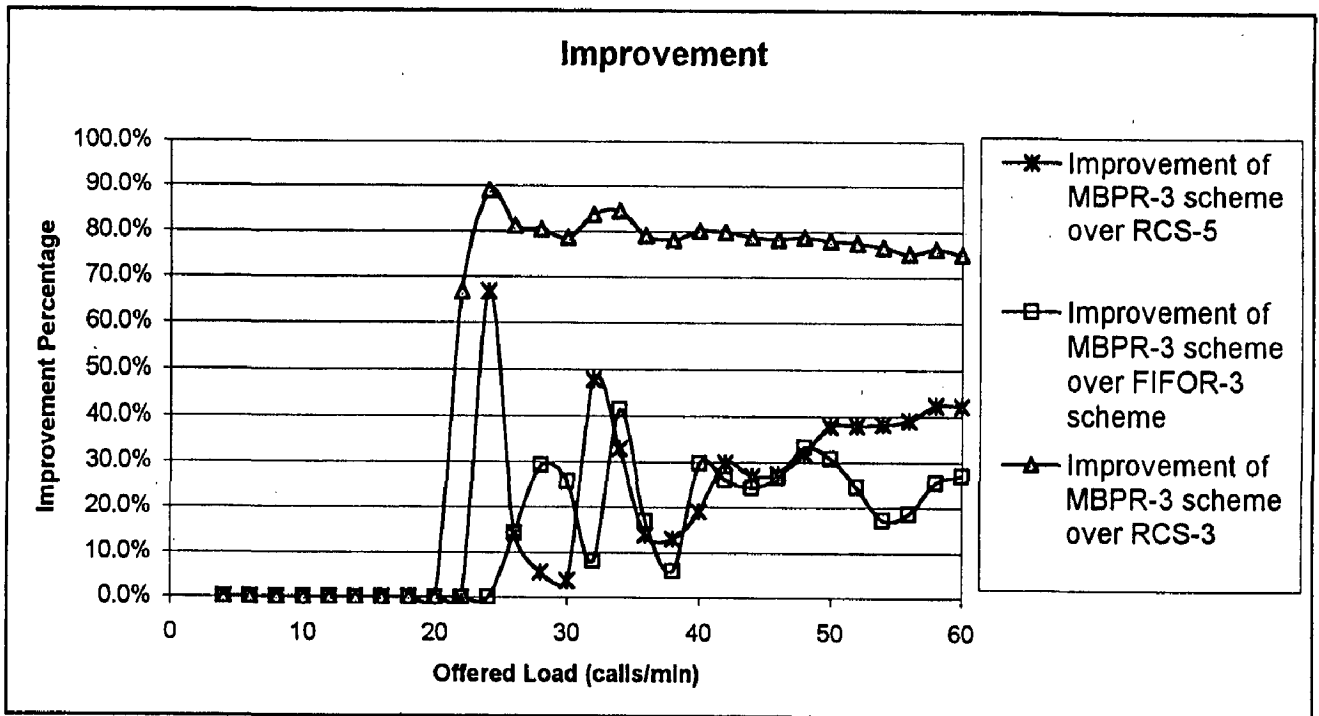


Fig.5.6. Improvement percentage of MBPR-3 scheme over RCS-5, FIFOR-3 scheme and RCS-3.

$$\text{Blocking Increase}(S_1, S_2) = \frac{(b(S_1) - b(S_2))}{S_1} * 100 \% \quad (5.2)$$

Where $b(S)$ is the blocking probability using scheme S .

Substitute various schemes for S_1 and S_2 .

In Fig. 5.7 there is a significant increase in the blocking probability in RCS-5 as compare to MBPR-3, the average increase is 21 %. In Fig. 5.6 we have seen that the MBPR-3 has an average improvement in decrease forced termination probability over RCS-5 of 19.7 %. This means that the hybrid scheme achieves significant improvement compared to reservation scheme.

MBPR-3 scheme has an average increase of blocking probability of 33.6 % as compared to MBPS, the curve in Fig. 5.7 shows that the blocking increase probability has a high rate of degradation as load increases, and has minimum a value of 16.3 %. The average increase in blocking probability of MBPR-3 scheme compared to RCS-3 is 1.5 % and its minimum value is -5 %. The negative value means that MBPR-3 scheme has less blocking probability for some points than RCS-3. It is clear that the increase of blocking probability is introduced by reservation scheme, not the hybrid scheme.

5.2.5 Carried Load

An important performance measure is the carried load, which is the ratio of number of calls the system can handle to the total traffic in the system.

Fig. 5.8 shows the carried load for handover schemes. When the traffic is low, all calls are handled in all schemes, as traffic increases, the carried load decreases. FSS has the least carried load. MBPR-3 and FIFOR-3 have the same carried load. Best carried load is achieved by MBPS and FIFO schemes.

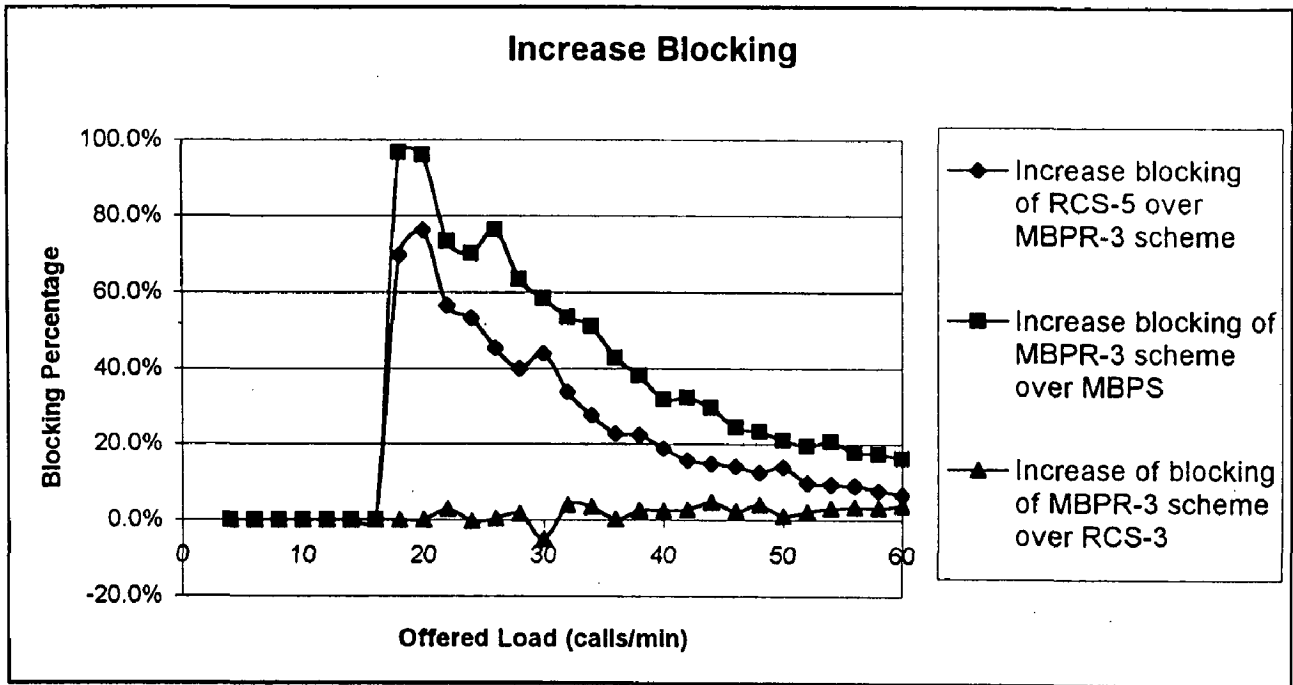


Fig. 5.7. Increase percentage of blocking probability.

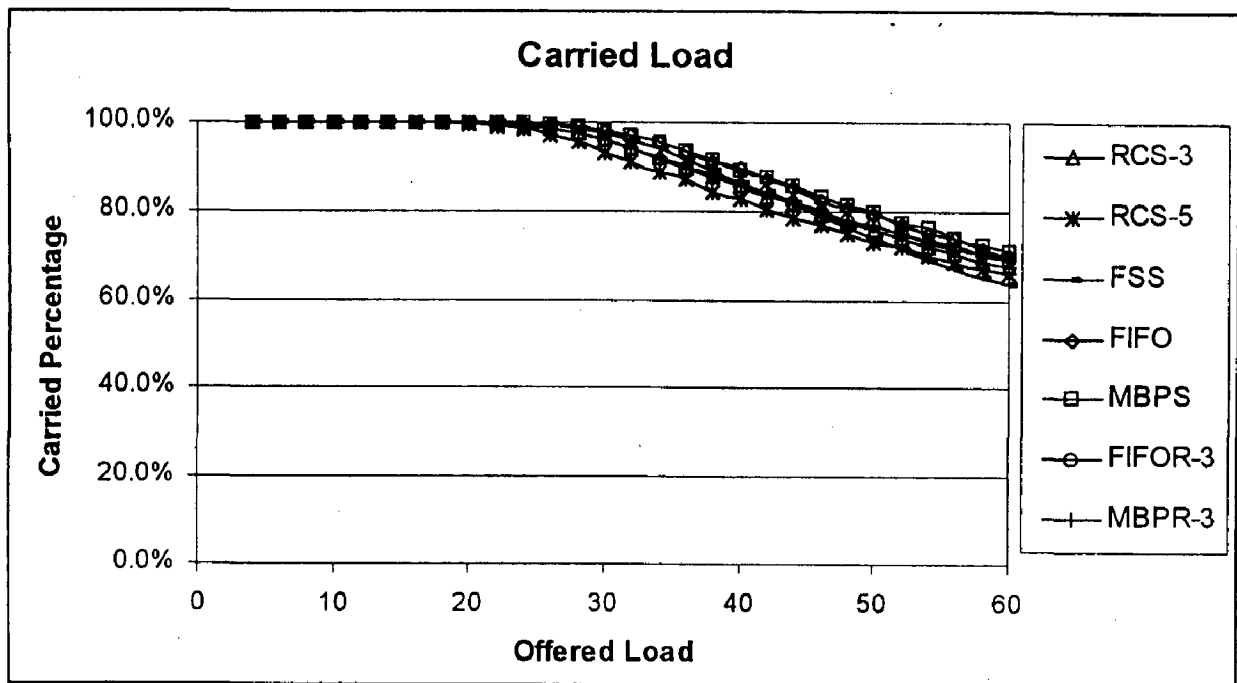


Fig. 5.8 Carried load for various handover schemes.

5.3 RESULTS IN SIMULATION MODEL II

In model II, the simulation was run using default simulation parameters. 100000 calls were sampled in one arbitrary cell of the simulation environment. Calls may require a fixed part of the network to complete their connections. The forced termination and blocking probability is expected to be higher than what was found in simulation model I, this because calls may require backbone link which might be engaged. Fig. 5.9 shows comparison of forced termination and blocking probabilities of MBPS scheme, using model I and model II.

The forced termination and blocking probability could be the same for model I and model II if both we apply sufficient bandwidth at backbone links. So calls and handovers fail only due to the lack of radio bandwidth.

5.3.1 Reserve Channels at Backbone Link

In model II, channels could be reserved at the backbone link as well as the radio link. Using reserved channels in both radio and backbone link lead to less forced termination probability as shown in Fig. 5.10. Channel reservation at ATM-based backbone link is valid as described in [14].

In Fig. 5.10 MBPS with 3 reserved radio channels and 5 reserved backbone channels (MBPS, RR 3, RB 5) has the least forced termination probability. FIFO scheme with 3 reserved radio channels and 5 reserved backbone channels (FIFO, RR 3, RB 5), has little improvement over (MBPS, RR 3, RB 3). Forced termination probability for (FIFO, RR 3, RB 3) is significantly higher than (FIFO, RR 3, RB 5). MBPS and FIFO with 3 reserved radio channels and zero reserved backbone channels have significant higher forced termination probability than the other described scheme. This shows the importance of using reserved channels on the backbone network links. There is significant improvement when MBPS queuing discipline is used over FIFO queuing discipline, this is clear when we use the same number of reserved channels on backbone link and radio link for both schemes.

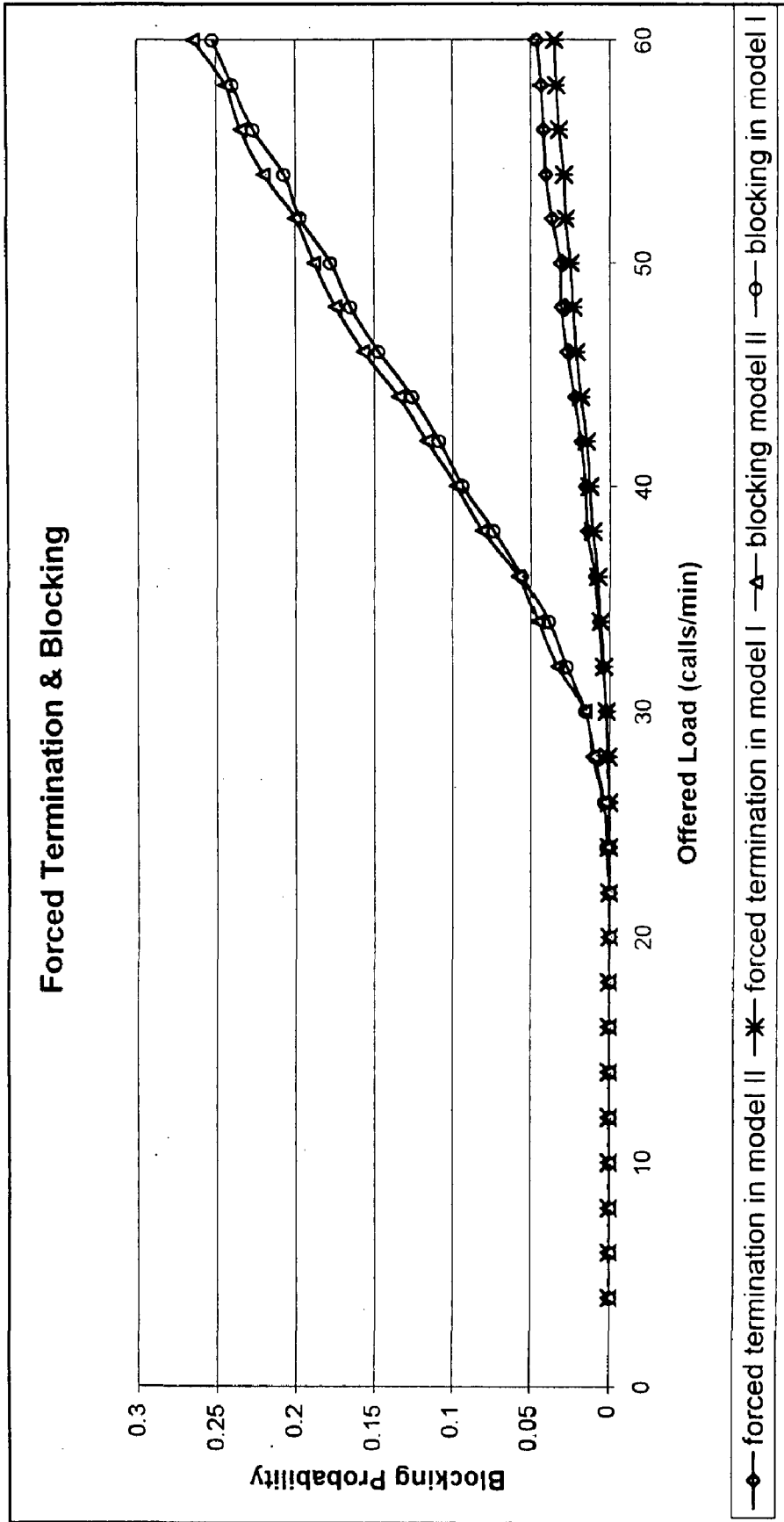


Fig. 5.9. Forced termination blocking probabilities in model I and model II for MBPS. Using default simulation parameters, and 90 channels for backbone link.

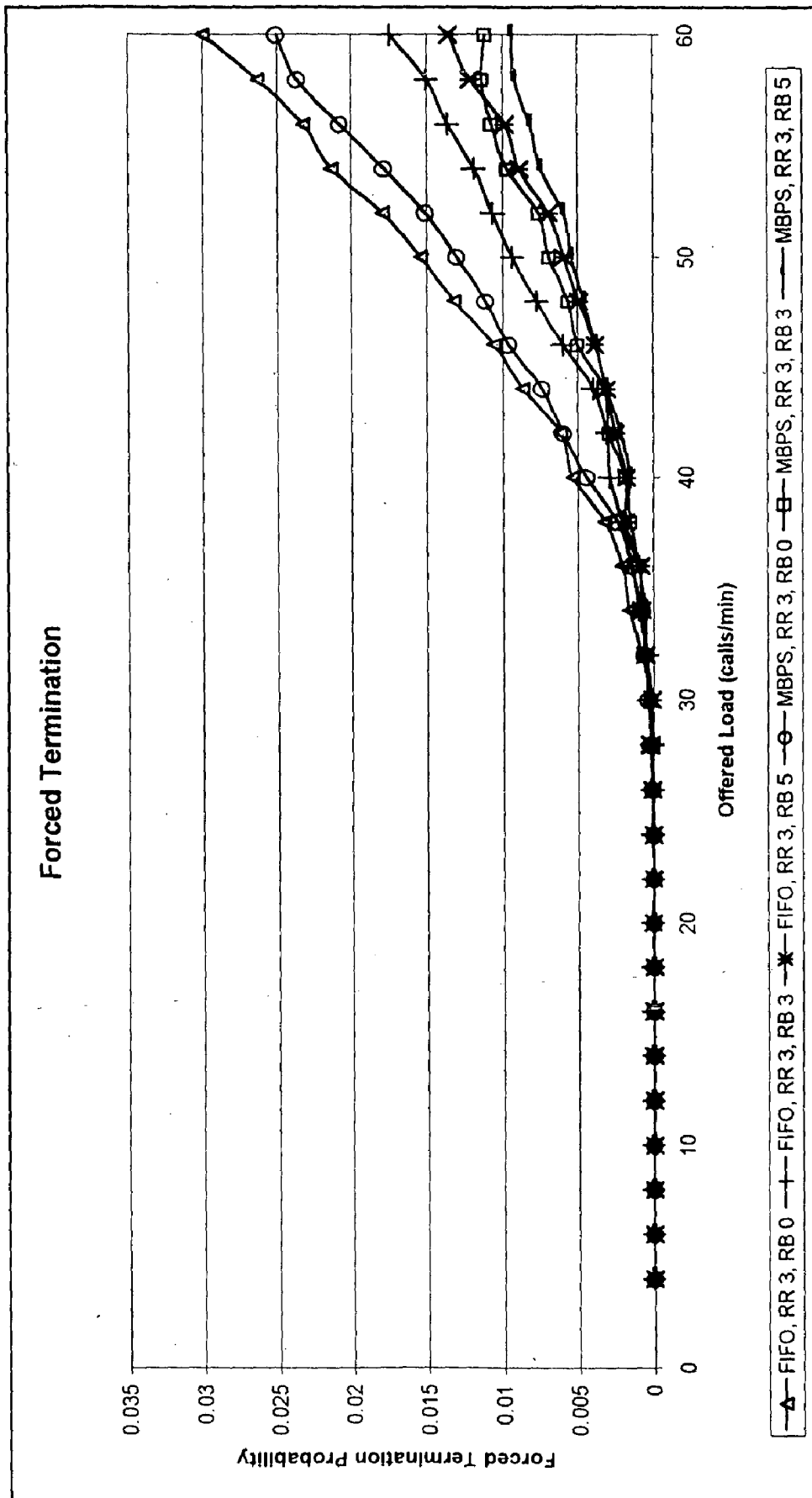


Fig. 5.10. Forced termination for hybrid scheme. Using default simulation parameters.

Fig. 5.11 gives the of blocking probability behavior in hybrid and non hybrid (FIFO and MBPS) schemes. It is clear that the hybrid schemes have more blocking probability. All hybrid schemes, which is basically queuing and reservation, approximately have the same blocking probability with minor differences.

5.3.2 Improvement of Reserving Channels at Backbone Link

Applying reservation scheme on backbone link leads to less forced termination probability, Fig. 5.12 explores the improvement in MBPS hybrid scheme (i.e. using reserved channels with MBPS), we gain by reserving number of channels on backbone link. The figure shows a significant improvement when 3 or 5 channels are reserved in backbone link. The maximum improvements are 58 % and 66 %, and the averages for these improvements are 27.2 % and 30.7 % respectively. The improvement percentage is positive for all points, which means applying reserved channels on backbone link always behaves better than non-reserved channels on backbone link

There is also an improvement when the number of reserved channels is increased at the backbone link. The average improvement in using 5 reserved channels instead of 3 reserved channels on backbone link is 6.8 %. Some improvement values are negative, the explanation is that for some points reserving 3 channels have less forced termination than using 5 channels, this occurred when the load is low, and the forced termination probability is quite small, but as the load increases, the improvement is clear. The general tendency of the graph is towards reducing the forced termination probability.

Similar results were obtained when comparing the same case for FIFO hybrid scheme. The average improvement is 22.9 %, 31.7 % and 14.2 % for 3 reserved channels, 5 reserved channels and using 5 reserved channels instead of 3 reserved channels on backbone link respectively.

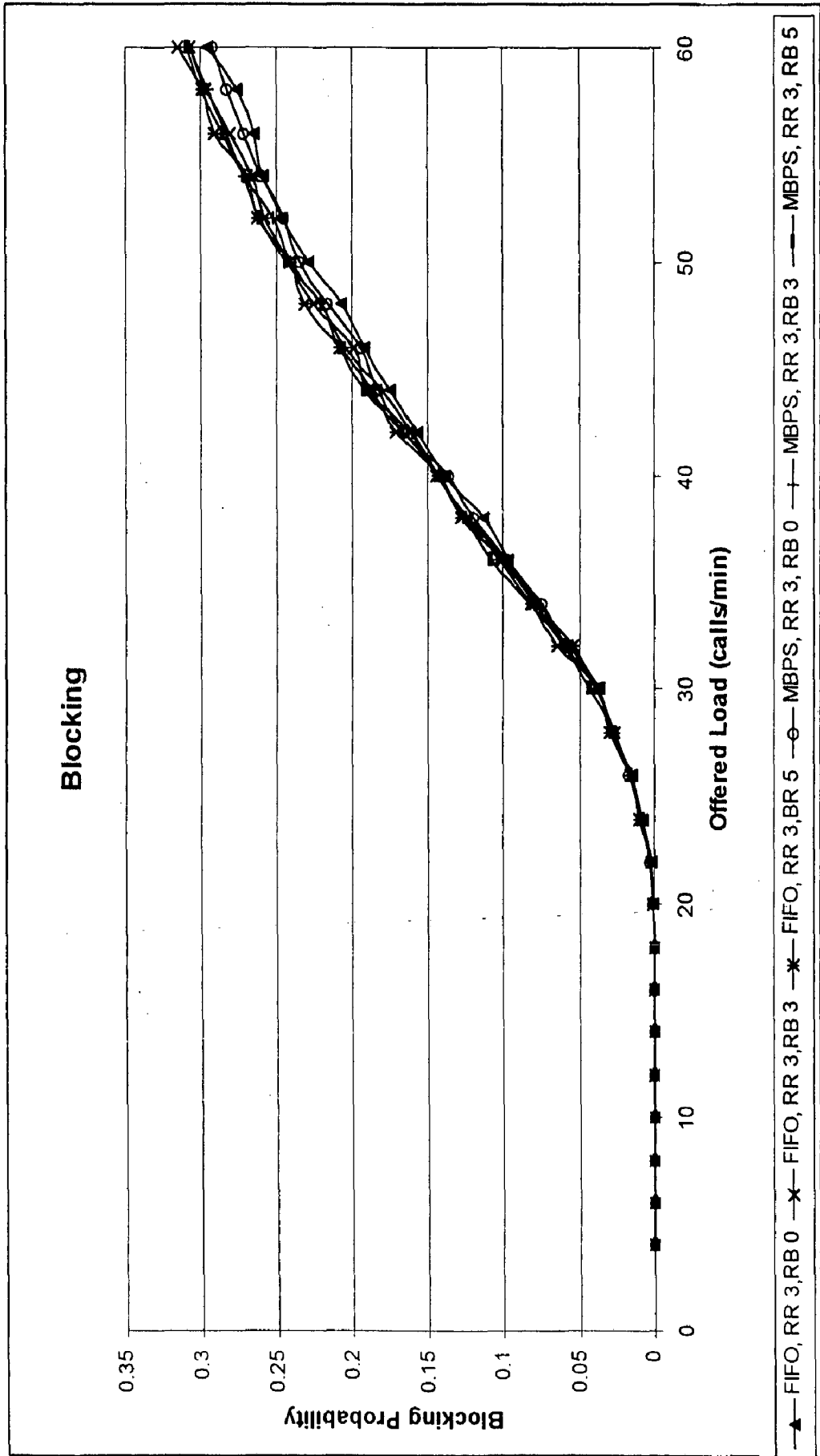


Fig. 5.11. Blocking probability in hybrid and non hybrid (FIFO, MBPS) schemes. Using default simulation parameters.

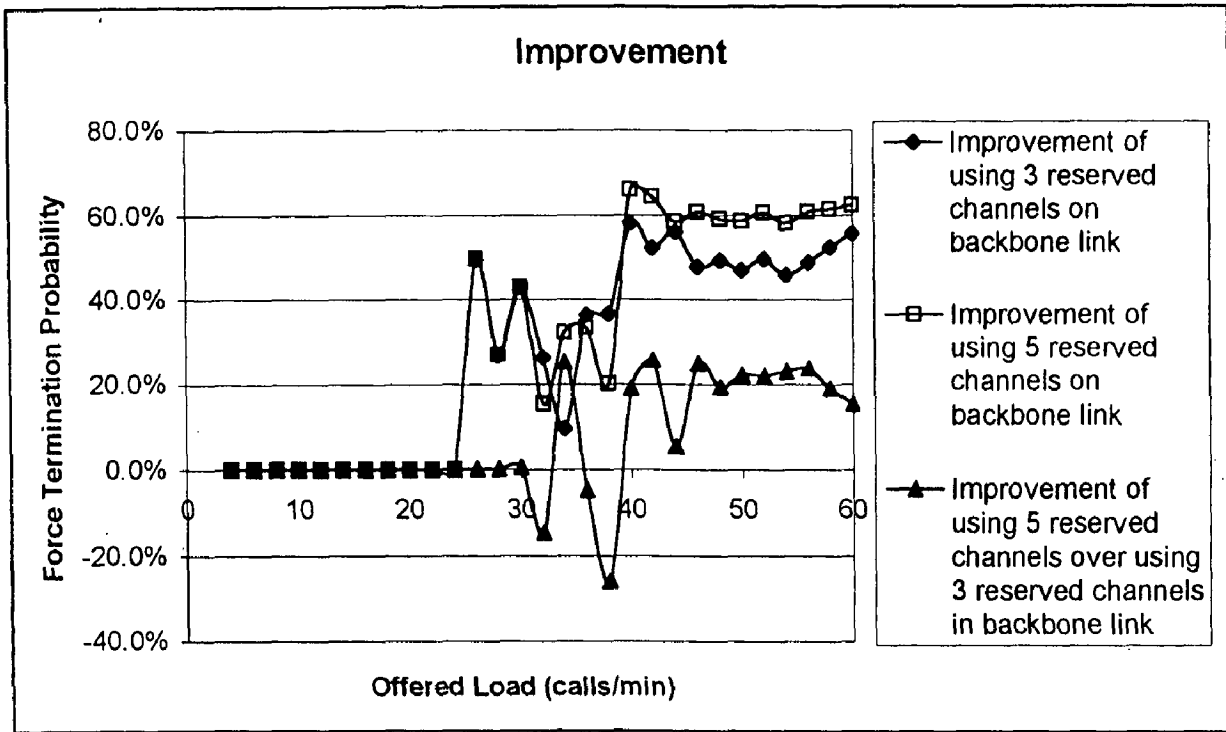


Fig. 5.12. Improvement in forced termination probabilities, gained by reserving channel on backbone link.

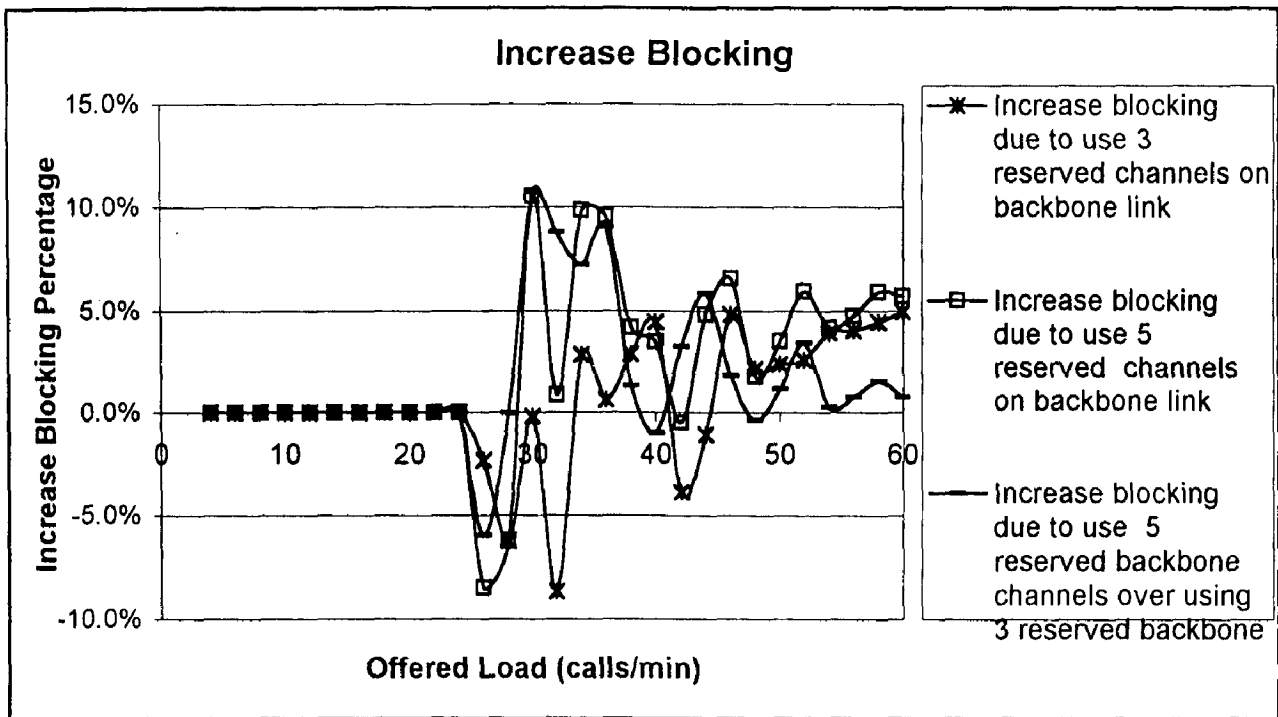


Fig. 5.13. Blocking increase Probability due to reserving channels on backbone link.

5.3.3 Increase in Blocking Due to Reserving Channels at Backbone Link

Reserving channels on backbone link, leads to slight increase of forced termination probability. Fig. 5.13 explores the increase in blocking probability in MBPS hybrid scheme (i.e. using reserved channels with MBPS). The average increase is 0.6 % and 2.3 % for using 3 and 5 reserved channels on backbone link, respectively.

The average increase due to use 5 reserved channels in backbone link compare to using 3 reserved channels on backbone link is 1.7 %. The results are similar of that were obtained when using FIFO hybrid scheme. The average blocking increase is 2.1 %, 4.9% and 2.8 % for 3 reserved channels, 5 reserved channels and using 5 reserved channels instead of 3 reserved channels on backbone link respectively.

5.3.4 Carried Traffic

Carried traffic versus the offered load is shown in Fig. 5.14, it is noticed that approximately all the schemes have the same performance. In moderate and high load MBPS and FIFO schemes have slightly better performance over other schemes (i.e. hybrid of reservation on the backbone and radio links and MBPS or FIFO).

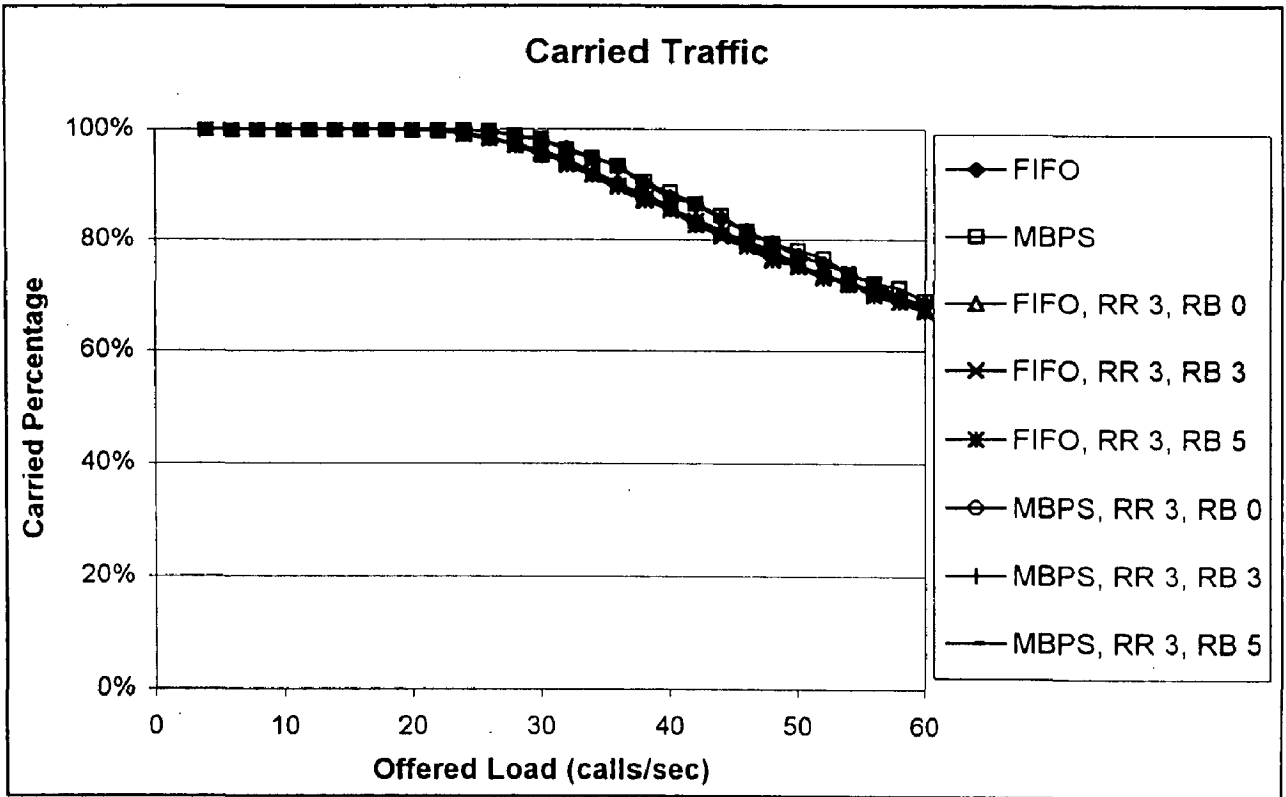


Fig. 5.14. Carried traffic for various handover schemes considering backbone network.

CONCLUSION

In this dissertation a new hybrid scheme for handover in ATM-based PCN is proposed. The scheme is quite simple. In literature, queuing handover requests and reserving a number of channels exclusively for handover request instead of blocking it is proposed. Both schemes work by giving handovers high priority over new calls. Hybrid scheme combines both queuing and reservation scheme, this gives handovers higher priority than queuing or reservation schemes.

From simulation, there is a significant improvement of the hybrid scheme over queuing or reservation scheme. However, there is also an increase of blocking probability in new calls. Reservation scheme always yields an increase in new calls blocking probability. In literature, many schemes for dynamic reservation are proposed as solution for this problem. This increase is inherited in the hybrid scheme. Also, however, the hybrid scheme achieves an important improvement over reservation scheme, in reducing both the blocking and the forced termination probabilities, i.e. improving QoS. Queuing discipline is important in improving QoS. We can draw the following recommendation to improve QoS:

- Queuing discipline that serve first the handover requests, which are about to be lost. FIFO is not a good candidate for that, MBPS and SPPQ [11] are good candidates for this job. However, queue discipline that depends on more measurement, and may be also on traffic patterns can lead to more accurate decision on which handover request should be served first.

Reservation is applied on both radio and backbone channels. It noticed that applying reservation scheme on backbone link leads to significant decrease in forced termination probability with a slight increase in blocking probability. This reflects the important of applying reservation scheme on both radio and backbone links. Applying dynamic version of reservation scheme may still leads to better results. Determining the number of reserved channels is also important and depends on the traffic patterns.

6.1 Scope For Future Work

- Implementing the hybrid scheme applying dynamic reservation channel scheme. By this an improvement in QoS is expected, in the sense of reducing the forced termination and blocking probabilities.
- Combination of this scheme with other schemes known scheme like channel borrowing and channel carrying may lead to more improvements in QoS.

REFERENCES

- [1] J.Schiller, "Mobile Communications", Great Britain: Addison-Wesley, 2000.
- [2] Y.C.Kim, D.E.Lee, B.J.Lee, Y.S.Kim and B.Mukherjee, "Dynamic Channel Reservation Based on Mobility in Wireless ATM Networks," *IEEE commun. Mag.*, pp. 47-51, Nov. 1999.
- [3] S.Tekinay and B.Jabbri, "A measurement-based prioritization scheme for handover in mobile cellular networks," *IEEE J.Select.Areas Commun.*, vol.10, no. 8, pp.1343-1350, 1992.
- [4] Katzela and M.Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive study," *IEEE Person.Communic.Mag.*, Jun. 1996.
- [5] J.Li, N.B. Shroff and E.K.p.Chong, "Channel carrying: A novel Handoff scheme for mobile cellular networks," *IEEE/ACM Trans. Networking*, vol. 7, no.1, Feb. 1999
- [6] "Wireless ATM: An Overview" available online at:
<http://www.ittc.ukans.edu/~prasiths/WirelessATM/content.htm>
- [7] C.P.Low "An efficient algorithm for the link allocation problem on ATM-Based personal communication networks," *IEEE J.select. areas in commu.*, vol.18, no. 7, July 2000
- [8] Averill M. Law and W.David Kelton, Simulation modeling and analysis, 2nd ed. New York: McGraw-Hill, 1991.

- [9] Tan Zhenhui "Mobility Management in Mobile ATM Network," Available online at:
http://www.telecomn.com/english/china_comm/FOCUS2_9910.htm
- [10] M.A. Maran, C. Chiasserini, R.L.Cigno and M.M.P. Di Torino "Local and global handovers for mobility management in wireless ATM networks," *IEEE Pers. commun. Mag.*, Oct. 1997.
- [11] H.G. Ebersman and O.K.Tonguz "Handoff ordering using signal prediction priority queuing in personal communication systems" *IEEE Trans. Veh. Technol.*, vol.48, no. 1, January 1999.
- [12] L. Ortigoza-Guerrero, and A.H. Aghvami "A prioritized handoff dynamic channel allocation strategy for PCS," *IEEE Trans. Veh. Technol.*, vol. 48, no. 4, January 1999.
- [13] M.D. Kulavaratharajah and A.H.Aghvami " Teletraffic performance evaluation of personal communication networks (PCN's) with prioritize handoff procedures," *IEEE Trans. Veh. Technol.*, vol.48, no.4, January 1999.
- [14] Oliver T.W. Yu and Victor.C.M Leung, "Adaptive resource allocation for prioritized call admission over an ATM-based wireless PCN," *J.Select. Areas Commun.*, vol. 15, no. 7, pp.1208-1225, Sep. 1997.
- [15] M. Ferracioli and R.Verdone "A general methodology based on the handover rate for network planning of cellular radio networks based on ATM," *IEEE J.select. areas in commu.*, vol.18, no. 7, July 2000.

APPENDIX

SOFTWARE LISTING

Queue.h

```
#include<iostream.h>
template<class T>
struct list{
T item;
int id;
list* next;
};

template<class T>
class queue{
protected:
    list<T>* front, *rear,*qptr;
public:
    int length,qid;
    queue();
    ~queue();
    int isEmpty();
    void add(T);
    int remove(T&);
    int del(int,int&);
    void go();
    void set(){qptr=rear;}
    void print();
};

template<class T>
queue<T>::queue(){
qptr=NULL;front=NULL;rear=NULL;
length=0;
}

template<class T>
queue<T>::~~queue(){
cout<<endl<<"queue destructor";
}
}
```

G10432



```

template<class T>
void queue<T>::add(T i){
list<T>* newptr=new list<T>;
newptr->item=i;
++length;
if(isEmpty()){
    front=rear=newptr;
    newptr->next=NULL;
    qid=0;
}
else {
    newptr->next=rear;
    rear=newptr;
    qid++;
}
newptr->id=qid;
}

```

```

template<class T>
int queue<T>::isEmpty(){
if( front==NULL)
    return 1; //yes
else return 0 ;//no
}

```

```

template<class T>
int queue<T>::remove(T& val){
list<T>*temp=rear;
if (isEmpty())
    return 0;
else{
    if (front==rear){
        front=rear=NULL;
    }
    else{
        while(temp->next!=front)
            temp=temp->next;
        front=temp;
        temp=temp->next;
        front->next=NULL;
    }
}
}

```

```

    val=temp->item;
    delete temp;
    --length;
return 1;
}

```

//implementation of del for queue class

```

template<class T>
int queue<T>::del(int ind,int&call_type){
list<T>*temp=rear;
list<T>* ptr=rear;
if (isEmpty())
    return 0;//delete fail
else{

    while( (ptr->id!=ind)&& (ptr!=NULL))
        ptr=ptr->next;
    }
    if(ptr==NULL) {cout<<endl<<"Fatal Error:delete fail id not exist";exit(1);}

    if( (ptr==rear) && (ptr==front) ){// 1) only one node
        ptr=front=rear=NULL;
    }

    else if(ptr==rear){ // 2) ptr points to the rear node
        rear=rear->next;
        ptr->next=NULL;
        ptr=NULL;
    }

    else if(ptr==front){ // 3) ptr points to the last node
        while(temp->next!=front)
            temp=temp->next;
        front=temp;
        temp=ptr;
        front->next=NULL;
        ptr=NULL;
    }

    else{ // 4) ptr points to the node in between
        while(temp->next!=ptr)
            temp=temp->next;
        temp->next=ptr->next;
    }
}

```

```
temp=ptr;
ptr=NULL;
temp->next=NULL;
}
```

```
call_type=temp->item.type;
delete temp;
--length;
return 1;
}
```

```
template <class T>
void queue<T>::go(){
if(qptr==NULL)
    qptr=rear;
```

```
if(qptr==front)
    qptr=rear;
else
    qptr=qptr->next;
}
```

```
template <class T >
void queue<T>::print(){
}
```

rand.h

```
#define MODULS 2147483647
#define MULT1 24112
#define MULT2 26143
/* set default seed for 100 streams */
static long zrng[] ={
193272912, 281629770,20006270,1280689831, 2096730329,1933576050
};

//generate next random
float rand (int stream)
{
long zi, lowprd, hi31;
zi =zrng[stream];
lowprd=(zi& 65535) * MULT1;
hi31 =(zi >> 16) * MULT1 + (lowprd >> 16);
zi  =((lowprd & 65535)- MODULS) + ((hi31 & 32767) << 16) + (hi31 >> 15);
if (zi<0) zi+=MODULS;
lowprd=(zi & 65535)*MULT2;
hi31 =(zi>>16) * MULT2 + (lowprd >> 16);
zi  =((lowprd & 65535) - MODULS ) + (( hi31 & 32767) << 16) + ( hi31 >> 15);
if (zi<0) zi+=MODULS;
zrng[stream] = zi ;
return ((zi >> 7 | 1) + 1) / 16777216.0;
}

// set the current zrng for stream "stream" to zset
void randst(long zset, int stream)
{
zrng[stream]= zset;
}

//return the current zrng for stream "stream"
long randgt(int stream)
{
return zrng[stream];
}
```

/*

SIMULATION MODEL I

SIMULATION PROGRAM FOR HANDOVER IN PERSONAL COMMUNICATION NETWORKS

Schemes simulated are:

- 1- FSS
- 2- RCS
- 3- FIFO
- 4- MBPS
- 5- FIFO + GUARD
- 6- MBPS + GUARD

*/

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
#include<fstream.h>
#include"mylib\queue.h"
#include"rand.h"
```

```
#define max_channels 30
```

```
#define MAX_SAMPLES 100000
```

```
#define mcl 60 //mean call length
#define hmcl 30 //handover mean call length
#define max_in_q 10 //average time in the queue
#define max_queue_size 999//
#define GUARD 3
```

```
#define NOW 0
#define NEWCALL 0
#define HANDOVER 1
#define RELEASE 2
```

```
#define FIFO 0
#define MBPS 1
#define GFIFO 2
```

```

#define GMBPS 3
#define RCS 4
#define FSS 5

```

```

struct calls{
    float atime;
    float end;
    int type;
};
struct handover{
    float atime;
    float priority;
    float q_time;
    int type;
};

```

```

class Random{
protected:
    float uniform(float, float);
    int int_uniform(int);
    float expon(float);
    int random_integer(float probab_dist[]);
};

```

```

template <class T>
class ho_queue: public queue <class T>{
public:
    T get(){return (qptr->item);}
    void del();
    void m_add(T);
};

```

```

template <class T>
void ho_queue<T>::m_add(T i){
{
    float val;
    list<T>* newptr=new list<T>;
    list<T>* temp=rear;
    newptr->item=i;
    val=i.q_time;
    ++length;
}
}

```

```

    if(isEmpty()){
        front=rear=newptr;
        newptr->next=NULL;
        qid=0;
    }
    else {
        // insert at the head or tail of the q
        if(rear->item.q_time<=val){
            newptr->next=rear;
            rear=newptr;
            qid++;
        }
        else if(front->item.q_time>val){
            front->next=newptr;
            front=newptr;
            front->next=NULL;
            qid++;
        }

        else{ //insert in between.
            while( (temp->next->item.q_time>val)&&(temp->next-
>next!=NULL) )

                temp=temp->next;
            newptr->next=temp->next;
            temp->next=newptr;
            qid++;
        }
    }

    newptr->id=qid;
} //else
}

```

```

template <class T>
void ho_queue<T>::del(){
    list<T>*temp=rear;

    if( (qptr==rear) && (qptr==front) ){// 1) only one node
        qptr=front=rear=NULL;
    }

    else if(qptr==rear){ // 2) qptr points to the rear node

```



```

    rear=rear->next;
    qptr->next=NULL;
    qptr=rear;
}

```

```

else if(qptr==front){ // 3) qptr points to the last node
    while(temp->next!=front)
        temp=temp->next;
    front=temp;
    temp=temp->next;
    front->next=NULL;
    qptr=rear;
}

```

```

else{ // 4) qptr points to the node in between
    while(temp->next!=qptr)
        temp=temp->next;
    temp->next=qptr->next;
    temp=qptr;
    qptr=qptr->next;
    temp->next=NULL;
}

```

```

delete temp;
--length;

```

```

}

```

```

template<class T>
class call_queue:public queue <class T>{
public:
    void get(int&id,T&call){id=qptr->id ; call=qptr->item;}

```

```

};

```

```

class Events:public Random{

```

```

public:

```

```

    float clock,next_call,next_handover,ho_delay,miat,hmiat;
    int busy_channels, next_event_type,max_q_len;
    int scheme;
    int call_id;
    long int total_calls,blocked,new_success,q_len,ho_success,

```

```

        ho_fail;
calls call;
call_queue<calls> call_list;
handover ho;
ho_queue<handover> q_ho;
Events();
void new_call();
void new_handover();
void release_channel(int);
private:
    void q_scan();

};

class simulation: public Events{
public:
    simulation();
    simulation(int);
    ~simulation(){fout.close();}
    void start();
    void start(long int);
    void traffic(float,int);
    void report();
    void save();
private:
    float load;
    ofstream fout;

    void timing();
    void intialize();

};

```

```

/*****
                                *MAIN PROGRAM*
*****/
void main(void){
simulation Fcell(FIFO);
simulation Mcell(MBPS);
simulation GFcell(GFIFO);
simulation GMcell(GMBPS);
simulation Ncell(FSS);//non priorities scheme
simulation Rcell(RCS);

int i;
float c;

clrscr();

/**/c=240;
cout<<"\n\n Simulation for MBPS ";
for( i=1;i<30;i++){
    Mcell.traffic(c,50);
    Mcell.start();
    Mcell.save();
    c+=120;
    cout<<"U";

} /**/

/**/c=240;
cout<<"\n\n Simulation for MBPS + GUARD scheme ";
for( i=1;i<30;i++){
    GMcell.traffic(c,50);
    GMcell.start();
    GMcell.save();
    c+=120;
    cout<<"U";

} /**/

/**/c=240;

```

```

cout<<"\n\n Simulation for FIFO scheme ";
for( i=1;i<30;i++){
    Fcell.traffic(c,50);
    Fcell.start();
    Fcell.save();
    c+=120;
    cout<<"Û";

}
/**/

/**/c=240;
cout<<"\n\n Simulation for FIFO + GUARD scheme ";
for( i=1;i<30;i++){
    GFcell.traffic(c,50);
    GFcell.start();
    GFcell.save();
    c+=120;
    cout<<"Û";

}
/**/

/**/c=240;
cout<<"\n\n Simulation for FSS scheme ";
for( i=1;i<30;i++){
    Ncell.traffic(c,50);
    Ncell.start();
    Ncell.save();
    c+=120;
    cout<<"Û";

}
/**/

/**/c=240;
cout<<"\n\n Simulation for          RCS scheme ";
for( i=1;i<30;i++){
    Rcell.traffic(c,50);
    Rcell.start();
    Rcell.save();
    c+=120;
    cout<<"Û";

```

```

    }
    /**/

cout<<"\n\n Simulation finished successfully, Press any key.....";
getch();

clrscr();

}
/*****/
/* Member functions implementation of Random class */
float Random::expon(float mean){
float u;
do{/**/u=rand(2);
}while(u==0);
float exp_u=-mean*log(u);
return exp_u;//-mean*log(u);
}

/* Member functions implementation of Events class */
Events::Events(){
next_call=0;
clock=0;
busy_channels=0;
blocked=0;
new_success=0;
ho_success=0;
q_len=0;
ho_delay=0;
ho_fail=0;
max_q_len=0;
total_calls=0;
call_list.length=0;
q_ho.length=0;
}
void Events::new_call(){
//schedule next call
next_call=expon(miat)+clock;
total_calls++;
if(busy_channels + GUARD <max_channels){
//free channel available
busy_channels++;
/*

```

the new arrive call can also make a handover request and free a channel. in both cases it end of a call.
 mcl for voluntary call termination is more than mcl for handover termination
 > and the rest of the new call have mcl larger.

```

*/
call.end=expon(mcl)+clock;
call.atime=clock;

if (call_list.length > max_channels) {
    cout<<"\n\n new call :calls more than channels";
    getch();
    exit(1);
}

call_list.add(call);//insert this call into a list of calls

new_success++; //no of new call that success to have a channel
}
else{// no free channel available, call blocked.
    blocked++;
}

}

void Events::new_handover(){
//schedule next handover.
next_handover=expon(hmiat) +clock;
total_calls++;
if(busy_channels <max_channels){
//serve the handover
//free channel available
busy_channels++;
/*
the handover recorded as new call with less mcl,
mean call length for handover is 'hmcl'
MU can continu roaming and move to other cell and issue new HO request.
*/
call.end=expon(hmcl)+clock;
call.atime=clock;
if (call_list.length > max_channels) {

```

```

        cout<<"\n\handover :calls more than channels";
        getch();
        exit(1);
    }
    call_list.add(call);//insert this call into a list of calls
    ho_success++; //HO request success to have a free channel
    ho_delay+=0;
}
else{//no free channel available

    if( (scheme==RCS)|| (scheme==FSS) ){
        ho_fail++;
    }

    else{
        //queue handover request.

        ho.atime=clock;
        ho.priority=0;
        ho.q_time=expon(max_in_q);

        if (q_ho.length > 999) {
            ho_fail ++;
        }

        if( (scheme==FIFO)|| (scheme==GFIFO) )
            q_ho.add(ho);
        else if( (scheme==MBPS)|| (scheme==GMBPS) )
            q_ho.m_add(ho);

        q_len++;
        if( max_q_len < q_ho.length ) max_q_len=q_ho.length;
    } // use queue scheme.
}

}

}

void Events::release_channel(int ind){

handover ho;
int ct;
call_list.del(ind,ct);

```

```

busy_channels--;
if(!q_ho.isEmpty())//there is HO request being queued
while( (!q_ho.isEmpty()) && (busy_channels<max_channels) )
{
//scan the queue to drop any old HO request
q_scan();
//assign priority based on MBPS or SPPQ scheme.

//take a HO call from the queue
if(!q_ho.isEmpty()){
q_ho.remove(ho);
busy_channels++;
/*the handover recorded as new call with less mcl,
mean call length for handover is 'hmcl'
MU can continu roaming and move to other cell and issue
new HO request.*/

call.end=expon(hmcl)+clock;
call.atime=clock;

if (call_list.length > max_channels) {
cout<<"\n Q handover :calls more than channels";
getch();
exit(1);
}

call_list.add(call);//insert this call into a list of calls
ho_success++; //HO request success to get a free channel
ho_delay+=clock-ho.atime; //delay for HO in queue
}
}
}

void Events::q_scan(){

handover ho;
int id, len;
q_ho.set();
len=q_ho.length;
for(int i=0;i<len/**/;i++){
ho=q_ho.get();
if ( (clock-ho.atime)> ho.q_time/*max_in_q/*expon(max_in_q)*/*/){
//HO will be dropped because it wait too much.
ho_fail ++;
}
}
}

```



```

        q_ho.del();
    }
    else
        q_ho.go();
}

}

/* Member functions implementation of Simulation class */
simulation::simulation(){

    fout.open("ho.txt");
}

simulation::simulation(int s){

    scheme=s;

    if(scheme==FIFO){
        fout.open("FIFO.TXT");
        if(!fout){
            cout<<"can't open FIFO.TXT";
            exit(0);
            getch();
        }
    }

    else if(scheme==MBPS){
        fout.open("MBPS.TXT");
        if(!fout){
            cout<<"can't open MBPS.TXT";
            exit(0);
            getch();
        }
    }

    else if(scheme==GMBPS){
        fout.open("GMBPS.TXT");
        if(!fout){
            cout<<"can't open GMBPS.TXT";
            exit(0);
            getch();
        }
    }
}

```

```

    }

else if(scheme==GFIFO){
    fout.open("GFIFO.TXT");
    if(!fout){
        cout<<"can't open MBPS.TXT";
        exit(0);
        getch();
    }
}

else if(scheme==FSS){
    fout.open("FSS.TXT");
    if(!fout){
        cout<<"can't open FSS.TXT";
        exit(0);
        getch();
    }
}

else if(scheme==RCS){
    fout.open("RCS.TXT");
    if(!fout){
        cout<<"can't open RCS.TXT";
        exit(0);
        getch();
    }
}

}

void simulation::start(){

next_call=NOW;
next_handover=NOW;
intialize();
while(total_calls<MAX_SAMPLES ){

    timing();
    switch(next_event_type){
        case NEWCALL : new_call();break;
        case HANDOVER: new_handover();break;
        case RELEASE : release_channel(call_id);break;
    }
}
}

```

```

        }
    }
}
void simulation::traffic(float l,int percentage ){

    load=l;

    /* float u[4],ui;
    */

    miat=(float(load)/100) * percentage;
    hmiat=(float(load)/100) * (100-percentage);

    miat=3600/miat;
    hmiat=3600/hmiat;
}

void simulation::report(){
}

void simulation::save(){

    long int processed= new_success+blocked+ho_success+ho_fail;
    float erlang=float(60*load)/3600;
    fout<<load<<' ';
    fout<<erlang<<' ';
    fout<<(float(blocked)/float(processed) ) <<' ';
    fout<<(float(ho_fail)/float(processed) ) <<' ';
    fout<<(float(ho_success + new_success) /float(processed) );
    fout<<endl;

}

void simulation::intialize(){
    Events::Events();
}

void simulation::timing(){
    float min_time_event=1.0e30;
    calls tmpcall;
    int id,len;
    next_event_type=RELEASE;
    call_list.set();
    len=call_list.length;
}

```

```

for(int i=0;i<len/*call_list.length*/;i++){
    call_list.get(id,tmpcall);
    if (tmpcall.end < min_time_event ){
        min_time_event=tmpcall.end;
        call_id=id;
    }
    call_list.go();
}
if(next_call<min_time_event){
    min_time_event=next_call;
    next_event_type=NEWCALL;
}
if(next_handover<min_time_event){
    min_time_event=next_handover;
    next_event_type=HANDOVER;
}
clock=min_time_event;
}

```

/*

SIMULATION MODEL II

SIMULATION PROGRAM FOR HANDOVER IN PERSONAL COMMUNICATION NETWORKS

Schemes simulated are:

- 1-FIFO
- 2-MBPS
- 3-FIFO + RESERVATION
- 4-MBPS + RESERVATION

*/

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
#include<fstream.h>
#include"mylib\queue.h"
#include"mylib\rand.h"

#define RADIO_CHANNELS      30
#define LOCALLINK_CHANNELS 30

#define MAX_SAMPLES 10000

#define mcl 60 //mean call length
#define hmcl 30 //handover mean call length
#define max_in_q 10 //average time in the queue
#define max_queue_size 999

#define NO_FREE_CHANNEL 0
#define CALL_ADMITTED 1

#define NO_OF_BTS 4

#define NOW 0

#define NEWCALL 0
#define HANDOVER 1
#define RELEASE 2
```

```
#define INCELL          0
#define INCLUSTER      1
#define OUTCLUSTER     2
```

```
#define NEWCALL 0
#define HANOVER 1
```

```
#define GUARD      3
#define BGUARD    5
```

```
#define FIFO 0
#define MBPS 1
#define GBFIFO 2
#define GBMBPS 3
```

```
struct calls{
    float atime;
    float end;
    int type;
};
struct handover{
    float atime;
    int type;
    float priority;
    float q_time;
    float RSS;
    float delta_t;
};
```

```
class Random{
protected:
    float uniform(float,float);
    int int_uniform(int);
    float expon(float);
    int random_integer(float prob_dist[]);
};
```

```
template <class T>
class ho_queue:public queue <class T>{
public:
```

```

    T get(){return (qptr->item);}
    void del();
    void m_add(T);

};
template <class T>
void ho_queue<T>::m_add(T i){
    {
        float val;
        list<T>* newptr=new list<T>;
        list<T>* temp=rear;
        newptr->item=i;
        val=i.q_time;
        ++length;
        if(isEmpty()){
            front=rear=newptr;
            newptr->next=NULL;
            qid=0;
        }
        else {
            // insert at the head or tail of the q
            if(rear->item.q_time<=val){
                newptr->next=rear;
                rear=newptr;
                qid++;
            }
            else if(front->item.q_time>val){
                front->next=newptr;
                front=newptr;
                front->next=NULL;
                qid++;
            }

            else{ //insert in between.
                while( (temp->next->item.q_time>val)&&(temp->next-
>next!=NULL) )

                    temp=temp->next;
                newptr->next=temp->next;
                temp->next=newptr;
                qid++;
            }
        }

        newptr->id=qid;
    }
}

```

```
    }//else  
}
```

```
template <class T>  
void ho_queue<T>::del(){  
    list<T>*temp=rear;  
  
    if( (qptr==rear) && (qptr==front) ){// 1) only one node  
        qptr=front=rear=NULL;  
    }  
  
    else if(qptr==rear){ // 2) qptr points to the rear node  
        rear=rear->next;  
        qptr->next=NULL;  
        qptr=rear;  
    }  
  
    else if(qptr==front){ // 3) qptr points to the last node  
        while(temp->next!=front)  
            temp=temp->next;  
        front=temp;  
        temp=temp->next;  
        front->next=NULL;  
        qptr=rear;  
    }  
  
    else{ // 4) qptr points to the node in between  
        while(temp->next!=qptr)  
            temp=temp->next;  
        temp->next=qptr->next;  
        temp=qptr;  
        qptr=qptr->next;  
        temp->next=NULL;  
    }  
    delete temp;  
    --length;  
}
```

```
template<class T>
```



```

class call_queue:public queue <class T>{
public:
    void get(int&id,T&call){id=qptr->id ; call=qptr->item;}

};
class ATMswitch;
class BackboneLink{
public:
    BackboneLink();
    int capacity;
    int atms[2];
};
BackboneLink::BackboneLink(){capacity=50;}

class Events:public Random{

public:
    float clock,next_call,next_event_time,next_handover,ho_delay,miat,hmiat;
    int busy_channels, next_event_type,max_q_len;
    int call_id;
    long int total_calls,blocked,new_success,q_len,ho_success,
        ho_fail;
    long int incell_success, incell_blocked, incluster_success,
        incluster_blocked,outcluster_success,outcluster_blocked;
    calls call;
    call_queue<calls> call_list;
    handover ho;
    ho_queue<handover> q_ho;
    Events();
    void new_call(int,ATMswitch*,BackboneLink*);
    void new_handover(int,ATMswitch*,BackboneLink*,int);
    void release_channel(int,int,ATMswitch*,BackboneLink*);
private:
    void q_scan();

};
class ATMswitch{
public:
    //int atmsid;// ATM switch id
    //int bb;
    int call_admission(BackboneLink[],int);
    void bblink_release(BackboneLink[]);

```

```

    /*routing routine,
       Return an array of integers containing the indexes
       of backbone links, and return whether the route feasible.
    */
    int route(ATMswitch*,BackboneLink[],int,int,int[]);

};
int ATMswitch::call_admission(BackboneLink* bblink,int request){
/*call routing routine
this routine return an array of ATM backbone links
*/
if(request == HANDOVER){
    if (bblink[0].capacity>0){//Backbone Channel available
        bblink[0].capacity--;
        return CALL_ADMITTED;
    }
    else
        return NO_FREE_CHANNEL;
}

else /* (request==NEWCALL)*/ {
    if (bblink[0].capacity>BGUARD){//Backbone Channel available
        bblink[0].capacity--;
        return CALL_ADMITTED;
    }
    else
        return NO_FREE_CHANNEL;
}
}

void ATMswitch::bblink_release(BackboneLink* bblink){
    bblink[0].capacity++;
}

class BaseStation: public Events{
public:
    int cellid;
    int neighbors[6];
    int atms;
    void traffic(int,int);
    void initialize(){Events::Events();}
};

```

```

class simulation{
public:
    simulation();
    simulation(int);
    ~simulation(){fout.close();}
    BaseStation BTS[4];
    ATMswitch atmswitch[2];
    BackboneLink bblink[1];
    void netTopology();
    void start(int);
    void start(long int);
    //void traffic(int,int);
    void report();
    void save();
private:
    float sclock; //simulation clock
    long int load ;
    int BTS_index,scheme;
    ofstream fout;
    //int call_id;

    void timing();
    void initialize();

};
/*****
                                MAIN PROGRAM
*****/
void main(void){
simulation Fcell(FIFO);
simulation Mcell(MBPS);

simulation BFcell(GBFIFO);
simulation BMcell(GBMBPS);

int i,c=240;
clrscr();

/**/ cout<<"\n\n Simulation for FIFO scheme ";
    for( i=1;i<30;i++){
        Fcell.start(c);
        c+=120;
        Fcell.save();

```

```

    cout<<i<<<"U";

}
/**/

/**/ c=120;
    cout<<"\n Simulation for MBPS scheme";
    for( i=1;i<30;i++){
        c+=120;
        Mcell.start(c);
        Mcell.save();
        cout<<i<<<"U";
    } /**/

/**/ c=120;
    cout<<"\n Simulation for BMBP scheme + GUARD scheme";
    for( i=1;i<30;i++){
        c+=120;
        BMcell.start(c);
        BMcell.save();
        cout<<i<<<"U";
    } /**/

/**/ c=120;
    cout<<"\n Simulation for BFIFO scheme + GUARD scheme";
    for( i=1;i<30;i++){
        c+=120;
        BFcell.start(c);
        BFcell.save();
        cout<<i<<<"U";
    } /**/

cout<<endl<<"\npress any key .....";
getch();
clrscr();

}

```

```

/*****
/* Member functions implementation of Random class */
float Random::expon(float mean){
float u;
do{/**/u=rand(2);
}while(u==0);
float exp_u=-mean*log(u);
return exp_u;/**-mean*log(u);
}

float Random::uniform(float a, float b){
float u;
u=rand(2);
return a+u*(b-a);
}
int Random::int_uniform(int a){
int u;
u=random(a);
return u;
}

int Random::random_integer(float prob_dist[]){
int i;
float u;

u=rand(1);
for(i=0;u>=prob_dist[i];++i);

return i;
}

/* Member functions implementation of Events class */
Events::Events(){
next_call=0;
clock=0;
busy_channels=0;
blocked=0;
new_success=0;
incell_success=0;

```

```

inccell_blocked=0;
incluster_success=0;
incluster_blocked=0;
outcluster_success=0;
outcluster_blocked=0;
ho_success=0;
q_len=0;
ho_delay=0;
ho_fail=0;
max_q_len=0;
total_calls=0;
// miat=100;
// hmiat=100;
call_list.length=0;
q_ho.length=0;
}

void Events::new_call(int atmsid,ATMswitch* atms,BackboneLink* bblink){

float d1=(float)1/7,d2=2*d1;
float call_type_prob[]={d1,d2,1.0};
//int BACKBONE_CHANNELS =bb[0].capacity;
//schedule next call for this Base Transceiver Station
next_call=expon(miat)+clock;
total_calls++;
//call type.
/*
INCELL: allocate only radio channel
INCLUSTER call: allocate radio channel and channel in local link
OUTCLUSTER:allocate radio channel, local link channel, and Backbone channel
*/

int call_type=random_integer(call_type_prob);
switch(call_type){
    case INCELL:{
        /*call admission control take place on the BTS only
        */
        if(busy_channels+GUARD<RADIO_CHANNELS){
            //free channel available,
            busy_channels++;//so, it allocate radio channel
            call.end=expon(mcl)+clock;
            call.atime=clock;
            call.type=INCELL;
        }
    }
}

```

```

if (call_list.length > RADIO_CHANNELS) {
    cout<<"\n\n new call :calls more than channels";
    getch();
    exit(1);
}
call_list.add(call);//both MT in same cell, so add the call again.

new_success++;//no of new call that success to have a channel
incell_success++;//incell_success++;
}
else{// no free channel available,call blocked.
    blocked++;//blocked++; //both MT are blocked.
    incell_blocked++;//incell_blocked++;
}
break;} //case 1: in cell

case INCLUSTER:{
    if(busy_channels+GUARD<RADIO_CHANNELS){
        //free radio channel available
        if(busy_channels<LOCALLINK_CHANNELS){//check for local link
            busy_channels++;
            call.end=expon(mcl)+clock;
            call.atime=clock;
            call.type=INCLUSTER;

            if (call_list.length > RADIO_CHANNELS) {
                cout<<"\n\n new call :calls more than channels";
                getch();
                exit(1);
            }
            call_list.add(call);//insert this call into a list of calls
            new_success++; //no of new call that success to have a channel
            incluster_success++;
        }
    }
    else{// no free channel available, call blocked.
        blocked++;
        incluster_blocked++;
    }

break;}//case 2: in cluster

```



```

switch(call_type){
    case INCELL:{
        if( (busy_channels<RADIO_CHANNELS) ){
            //free radio channel available

            //serve the handover
            busy_channels++;
            /*
            the handover recorded as new call with less mcl,
            mean call length for handover is 'hmcl'
            Mt can continue roaming and move to other cell and issue new
            HO request.
            */
            call.end=expon(hmcl)+clock;//
            call.atime=clock;
            call.type=INCELL;
            if (call_list.length > RADIO_CHANNELS) {
                cout<<"\n\handover :calls more than channels";
                getch();
                exit(1);
            }
            call_list.add(call);//inster this call into a list of calls
            ho_success++; //HO request success to have a free channel
            outcluster_success++;
            ho_delay+=0;
            //}
        }
        else
        { //no free channel available
            //queue handover request.

            ho.atime=clock;
            ho.type=INCELL;
            ho.priority=0;
            ho.q_time=expon(max_in_q);

            if (q_ho.length > 999) {
                ho_fail ++;
            }

            if ((scheme==FIFO)||(scheme==GBFIFO))
                q_ho.add(ho);
        }
    }
}

```

```

else if( (scheme==MBPS)|| (scheme==GBMBPS))
    q_ho.m_add(ho);

q_len++;
if( max_q_len < q_ho.length ) max_q_len=q_ho.length;
}

```

```
break;} //case 1: in cell
```

```

case INCLUSTER:{
    if( (busy_channels<RADIO_CHANNELS) ){
        //free radio channel available

        //serve the handover
        busy_channels++;
        /*
        the handover recorded as new call with less mcl,
        mean call length for handover is 'hmcl'
        Mt can continue roaming and move to other cell and issue new
        HO request.
        */
        call.end=expon(hmcl)+clock;
        call.atime=clock;
        call.type=INCLUSTER;
        if (call_list.length > RADIO_CHANNELS) {
            cout<<"\n\handover :calls more than channels";
            getch();
            exit(1);
        }
        call_list.add(call);//insert this call into a list of calls
        ho_success++; //HO request success to have a free channel
        outcluster_success++;
        ho_delay+=0;
        //}
    }
    else
        { //no free channel available
        //queue handover request.

        ho.atime=clock;
        ho.type=INCLUSTER;
        ho.priority=0;
        ho.q_time=expon(max_in_q);

```

```

if (q_ho.length > 999) {
    ho_fail ++;
}

if ((scheme==FIFO)|| (scheme==GBFIFO))
    q_ho.add(ho);
else if( (scheme==MBPS)|| (scheme==GBMBPS))
    q_ho.m_add(ho);

q_len++;
if( max_q_len < q_ho.length ) max_q_len=q_ho.length;
}

```

```
break;}//case 2: in cluster
```

```

case OUTCLUSTER:{
    if( (busy_channels<RADIO_CHANNELS)&&
        (atms[atmsid].call_admission(bblink,HANDOVER)) ){
        //free radio channel available
        //send signal to ATM mobile switch
        /*
            serve the handover
            free channel available */
        busy_channels++;
        /*
            the handover recorded as new call with less mcl,
            mean call length for handover is 'hmcl'
            Mt can continu roaming and move to other cell and issue new
            HO request.
        */
        call.end=expon(hmcl)+clock;
        call.atime=clock;
        call.type=OUTCLUSTER;
        if (call_list.length > RADIO_CHANNELS) {
            cout<<"\n\handover :calls more than channels";
            getch();
            exit(1);
        }
        call_list.add(call);//insert this call into a list of calls
        ho_success++; //HO request success to have a free channel
        outcluster_success++;
    }
}

```

```

        ho_delay+=0;
        //}
    }
    else
    { //no free channel available
    //queue handover request.

    ho.atime=clock;
    ho.type=OUTCLUSTER;
    ho.priority=0;
    ho.q_time=expon(max_in_q);

    if (q_ho.length > 999) {
        ho_fail ++;
    }

    if ((scheme==FIFO)|| (scheme==GBFIFO))
        q_ho.add(ho);
    else if ( (scheme==MBPS)|| (scheme==GBMBPS))
        q_ho.m_add(ho);

    q_len++;
    if( max_q_len < q_ho.length ) max_q_len=q_ho.length;
    }
    break;} //case 3: out cluster
} //switch case
}

```

```

void Events::release_channel(int ind,int atmsid,ATMswitch*
atms,BackboneLink*bblink){

```

```

handover ho;
int BBavailabel;

```

```

int call_type,bb=1;
call_list.del(ind,call_type);
busy_channels--;
if(call_type==OUTCLUSTER){ //free channel at the backbone link
    atms[atmsid].bblink_release(bblink);
}

```

```

//scan the queue to drop any old HO request
q_scan();

```

```

if(!q_ho.isEmpty())//there is HO request being queued
while( (!q_ho.isEmpty()) && (busy_channels<RADIO_CHANNELS)&&(bb)
)
{
if( atms[atmsid].call_admission(bblink,HANDOVER) ){
//take a HO call from the queue
//backbone channel available
q_ho.remove(ho);
busy_channels++;

/*the handover recorded as new call with less mcl,
mean call length for handover is 'hmcl'
MT can continue roaming and move to other cell and issue
new HO request.*/

call.end=expon(hmcl)+clock;
call.atime=clock;
call.type=OUTCLUSTER;

if (call_list.length > RADIO_CHANNELS) {
cout<<"\n Q handover :calls more than channels";
getch();
exit(1);
}

call_list.add(call);//insert this call into a list of calls
ho_success++; //HO request success to get a free channel
ho_delay+=clock-ho.atime; //delay for HO in queue
}
else { /*ho_fail++*/;bb=0;}
}
}

```

```

void Events::q_scan(){

handover ho;
int id, len;
q_ho.set();
len=q_ho.length;
for(int i=0;i<len/**/;i++){
ho=q_ho.get();

```

```

    if ((clock-ho.atime)> ho.q_time/*max_in_q*/expon(max_in_q)*/){
        //HO will be dropped because it wait too much.
        ho_fail ++;
        q_ho.del();
    }
    else
        q_ho.go();
}
}

```

```

/* Member functions implementation of Simulation class */
simulation::simulation(){

```

```

    fout.open("ho.txt");
}

```

```

simulation::simulation(int s){
    scheme=s;

```

```

    if(scheme==FIFO){
        fout.open("BFIFO.TXT");
        if(!fout){
            cout<<"can't open BFIFO.TXT";
            exit(0);
            getch();
        }
    }

```

```

    else if(scheme==MBPS){
        fout.open("BMBPS.TXT");
        if(!fout){
            cout<<"can't open BMBPS.TXT";
            exit(0);
            getch();
        }
    }

```

```

    else if(scheme==GBMBPS){
        fout.open("GBMBPS.TXT");
        if(!fout){
            cout<<"can't open GBMBPS.TXT";
            exit(0);

```

```

        getch();
    }
}

else if(scheme==GBFIFO){
    fout.open("GBFIFO.TXT");
    if(!fout){
        cout<<"can't open GBFIFO.TXT";
        exit(0);
        getch();
    }
}

}

void simulation::netTopology(){

    BTS[0].neighbors[0]=1;
    BTS[0].neighbors[1]=2;
    BTS[0].neighbors[2]=3;
    BTS[0].neighbors[3]=3;
    BTS[0].neighbors[4]=3;
    BTS[0].neighbors[5]=3;

    BTS[1].neighbors[0]=0;
    BTS[1].neighbors[1]=2;
    BTS[1].neighbors[2]=3;
    BTS[1].neighbors[3]=2;
    BTS[1].neighbors[4]=2;
    BTS[1].neighbors[5]=2;

    BTS[2].neighbors[0]=0;
    BTS[2].neighbors[1]=1;
    BTS[2].neighbors[2]=3;
    BTS[2].neighbors[3]=1;
    BTS[2].neighbors[4]=1;
    BTS[2].neighbors[5]=1;

    BTS[3].neighbors[0]=1;
    BTS[3].neighbors[1]=2;
    BTS[3].neighbors[2]=0;
    BTS[3].neighbors[3]=0;
    BTS[3].neighbors[4]=0;
    BTS[3].neighbors[5]=0;
    BTS[0].atms=0;
}

```

```

    BTS[1].atms=0;
    BTS[2].atms=1;
    BTS[3].atms=1;
    bblink[0].atms[0]=0;
    bblink[0].atms[1]=1;
    bblink[0].capacity=90;

}

void simulation::start(int l){
int atmsid; //ATM Switch ID
for (int i=0;i<NO_OF_BTS;i++){
    BTS[i].next_call=NOW;
    BTS[i].next_handover=NOW;
    BTS[i].traffic(l,50);
    BTS[i].initialize();
    load=l;
}

netTopology();
while(BTS[0].total_calls<MAX_SAMPLES){

    timing();
    atmsid=BTS[BTS_index].atms;
    switch(BTS[BTS_index].next_event_type){
        case NEWCALL : BTS[BTS_index].new_call(atmsid,atmswitch,bblink);
            break;
        case HANDOVER:
BTS[BTS_index].new_handover(atmsid,atmswitch,bblink,scheme);
            break;
        case RELEASE :
BTS[BTS_index].release_channel(BTS[BTS_index].call_id,atmsid,atmswitch,bblink);br
eak;
    }

}

}

void BaseStation::traffic(int l,int percentage ){
    miat=(float(l)/100) * percentage;
    hmiat=(float(l)/100) * (100-percentage);

    miat=3600/miat;

```



```

    hmiat=3600/hmiat;
}
void simulation::report(){

}
void simulation::save(){
long int new_success=0, blocked=0, ho_success=0, ho_fail=0;//
for (int i=0;i<NO_OF_BTS;i++){
    new_success+=BTS[i].new_success;
    blocked+=BTS[i].blocked;
    ho_success+=BTS[i].ho_success;
    ho_fail+=BTS[i].ho_fail;

}

long int processed= new_success+blocked+ho_success+ho_fail;

float erlang=float(load *60)/3600;

fout<<load<<' ';
fout<<erlang<<' ';
fout<<(float(blocked)/float(processed) ) <<' ';
fout<<(float(ho_fail)/float(processed) ) <<' ';
fout<<(float(ho_success + new_success) /float(processed) );
fout<<endl;
}

void simulation::initialize(){
    //Events::Events();
}
void simulation::timing(){
    float min_time_event;
    calls tmpcall;
    int id,len;
    BTS_index=0;

for(int i=0;i<NO_OF_BTS;i++){
    min_time_event=1.0e30;
    BTS[i].next_event_type=RELEASE;
    BTS[i].call_list.set();
    len=BTS[i].call_list.length;

    for(int k=0;k<len/*call_list.length*/;k++){

```

```

BTS[i].call_list.get(id,tmpcall);
if (tmpcall.end < min_time_event ){
    min_time_event=tmpcall.end;
    BTS[i].call_id=id;
    BTS[i].next_event_time=min_time_event;

}
BTS[i].call_list.go();
}

if(BTS[i].next_call<min_time_event){
    min_time_event=BTS[i].next_call;
    BTS[i].next_event_type=NEWCALL;
    BTS[i].next_event_time=min_time_event;
}
if(BTS[i].next_handover<min_time_event){
    min_time_event=BTS[i].next_handover;
    BTS[i].next_event_type=HANDOVER;
    BTS[i].next_event_time=min_time_event;
}

} //for loop
min_time_event=1.0e30;
for( i=0;i<NO_OF_BTS;i++){

    if(BTS[i].next_event_time<min_time_event){
        min_time_event=BTS[i].next_event_time;
        BTS_index=i;
    }
}
sclock=min_time_event;
BTS[BTS_index].clock=sclock;
}

```