Improved Simulation System Performance for Wireless Communications using Efficient Multi-Threading Architectures

P.M.PAPAZOGLOU[1], D.A.KARRAS[2], R.C.PAPADEMETRIOU[3]
[1] Dept. of Informatics & Computer Technology, Lamia Institute of Technology, Greece & University of Portsmouth, United Kingdom
[2] Dept. of Automation Engineering, Chalkis Institute of Technology, Greece

[3] Dept. of Electronic & Computer Engineering, University of Portsmouth, United Kingdom

Abstract - Simulation systems play a major role in the design and evaluation of any experimental wireless network. The goal of this paper is to demonstrate that simulation model architectures affect simulation behavior, concerning network performance metrics, essentially and therefore, the optimal architecture should be investigated in order to achieve the most accurate and reliable results. It is found that the most critical components that determine simulation model performance are simulation time, network event scheduling and grade of concurrency. It is, also, found that simulation time in relation to event occurrence in the real network along with the usage of modern architectural concepts such as multi-threading technology constitute critical issues too in the development of an efficient simulation system for wireless communications. In order to evaluate the above findings an extensive experimental study has been conducted testing several discrete event simulation systems towards presenting the relation between thread modeling selections, simulation time and network performance.

Key-words: - cellular network, simulation model, multi-threading

1 Introduction

1.1 Simulating Wireless networks

Several systems for simulating wireless networks have been introduced in the literature. The adaptation of the simulation systems to the real network behavior is a major goal. Thus, the simulation has to be as realistic as possible. The Physical time and the event occurrence in a real network have to be reflected realistically inside the simulation model. When two or more events are happening at the same time, the concurrency is the most suitable methodology to model them. The simulation model architecture along with programming language is the most critical selection for the simulation system developer. In this paper, experimental discrete event simulation systems are tested and proposed towards designing a more realistic simulation environment. Moreover, the relation between thread selections, scheduling mechanisms, simulation time and network performance for designing and evaluating cellular communications are also presented. The basic Mobile User (MU) services (events) that are supported by a cellular network are:

- New call admission
- Reallocation (handoff)
- User movement
- Call Termination

The simulation system consists of four major components that can be categorized as follows:

- MU services model
- Operational parameters selection (e.g. number of cells, Base station positions, channel allocation scheme, etc)
- Mathematical models integration (propagation models, statistical distributions, signal computations, etc)
- Simulation time modeling. It should be noticed that simulation of event occurrence over simulation time has to reflect realistically the physical time of the network under investigation.

Finally, the MUs services have to be simulated based on the above simulation system components.

1.2 Discrete Event Simulation (DES)

Discrete Event Simulation [1-10] represents the most known simulation methodology especially for communication systems. According to DES concept, events are happening at discrete points in time within the simulation time. Simulation time is moving forward based on the event sequence. These events represent the basic physical network activities such as new call admission, etc. Each event is generated with a time stamp that is used for the event execution at a later time. The event occurrence over simulation time is defined by a scheduler [11-15] that selects events with minimum time stamp (maximum priority). Thus, the whole scheduling procedure is based on a priority queue. DES systems can be categorized as sequential (SDES) or parallel (PDES). SDES systems such as ns-2 [16,17] are the most popular among scientific community. In such a system, the scheduling mechanism can be analyzed in a three step cycle:

- Dequeue: Removal of an event with the minimum time stamp from the queue
- Execute: processing of the dequeued event
- Enqueue: Insertion of a new generated event in the queue

In ns-2 [16], only one event can be executed at any given time. If two or more events are scheduled to take place (to execute) at the same time, their execution is performed based on the first scheduled – first dispatched manner and so the real network behaviour can not be reflected realistically.

The PDES systems offer significant opportunities for speeding up the execution time of a complex network structure. In such systems, several additional issues due to multiple processing units existence, have to be faced effectively such as processor synchronization [18-20], load balance [21,18], etc. These systems do not change the concept of the event scheduling mechanism.

2 DES System Development

2.1 DES features

An efficient DES system has to offer several features and to satisfy some critical conditions. These features and conditions can be interpreted as factors that affect the DES system behavior. The above factors are summarized as follows:

- Modeling of the MUs services
- Interpretation and implementation of the real time to simulation time
- Scheduling mechanism (the most prominent of them being the Calendar Queue (CQ) scheduling)
- Concurrency

Time in general is divided in three categories [22]:

- Physical time (real time of the real network)
- Wall-clock time (execution time)
- Simulation time

According to [22], "***Simulation time is defined as a totally ordered set of values where each value represents an instant of time in the physical system being modelled…***". A major goal of a DES system is the realistic representation of the physical time into simulation time as well as the more realistic scheduling approach.

2.2 Calendar Queue (CQ) scheduling

CQ was first introduced by Brown R. This method, constitutes the most known scheduling mechanism among the most popular DES systems such as ns-2(Berkeley)[23], Ptolemy II (Berkeley) [24], Jist (Cornel University, USA)[25], etc. Performance improvements for CQ can be found in [26-28]. Each event is associated with a time stamp that defines its priority (in execution sequence). A CQ can be implemented as an array of lists where each list contains future events. A list of N events is partitioned to M shorter lists called Buckets that correspond to a specific time range. Using eq.(1), the bucket number m(e) where an event e will occur at time t(e) can be calculated

$$m(e) = \left\lfloor \frac{t(e)}{\delta} \right\rfloor \bmod M \qquad (1)$$

Where δ is a time resolution related constant. Let M=8, N=10, δ=1 and t(e)=3.52 (fig.1) for a new event e. Using eq.(2), the bucket number for event e is m(e)=3.

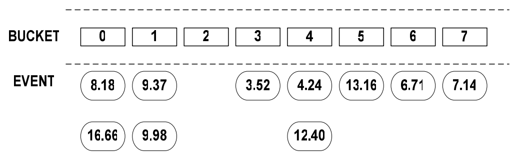

Fig. 1. A CQ operation

2.3 Multi-Threading Technology (MT)

A usual capability of a modern operating system (OS) is the execution of different programs (applications) at the "same time". Real application execution at the same time requires at least a system with N=P, where N are the applications and P the available processors. In most cases, only one processor is available and so the CPU time has to be shared between the running applications. This execution mechanism is called threading (multi-threading-MT). The traditional scheduling mechanism can not reflect realistically enough a simultaneous event occurrence as a concurrent (multi-threading) mechanism does.

2.3.1 The JVM Example

One of the most popular features of Java is the support of native multi-threading. The JVM controls the MT environment. More information for multi-threading capabilities of Java can be found in [29-31]. The OS faces JVM as a single application but within the JVM, multiple Java applications and/or multiple parts (segments) of one application can be executed (fig.2) concurrently. In a single processor machine, the active threads are executed with a high speed switching between them and so the impression of parallel execution is given. Through available methods of JVM, the priority level (1 to 10) of each thread can be defined. When the thread priorities are equal, the CPU time is distributed in a fair manner. The OS gives a time slice to JVM, and this time is re-distributed to java applications/threads inside the JVM environment. The event scheduling is also controlled by the JVM. This mechanism defines the real-time order of thread execution and can be categorized as non-preemptive or preemptive. The current thread is running forever and has to inform the scheduler explicitly if it is safe to start another waiting thread according to non-preemptive scheduling. In preemptive, a thread is running for a specific CPU time-slice and then the scheduler "preempts" it, (calling suspend()), and resumes another thread for the next available time-slice. In JVM, the execution time of every thread (in case of equal priorities) is balanced. Figure 2, illustrates three active threads that share a single processor. The sleep() method deactivates temporarily the current thread in order to give time for execution of another thread.

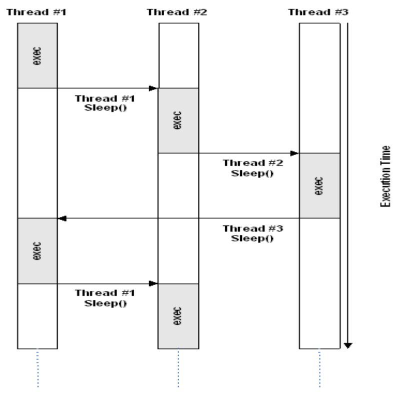

Fig. 2 Thread switching

2.3.2 MT scheduling

When the time slice given by JVM or by programmer to each thread (network event) execution is less than the required computational time, an event interleaving is achieved. This technique reflects more realistically the user competition for accessing common radio resources. Moreover, when a user is under processing, the simulation time flows also for the next upcoming user. Thus, the network decisions are more sophisticated and optimized to more user requests.

3 Experimental DES Models

3.1 Network modeling

All the experimental models are based on the concurrent concept that is implemented through Multi-threading. Due to the nature of the event occurrence in the real network, concurrency offers a chance to model more realistically the physical activities of the network. Three different architectures with increasing grade of concurrency have been implemented and tested in order to investigate the dependence of the results and simulation system performance from the multi-threading usage. The simulation models support the four basic services for the MUs as mentioned before. The simulation model operation is mainly focused in channel allocation procedure which is strongly connected with new call admission and reallocation (hand off). This procedure is also used in the case of an MU movement. Three conditions must be fulfilled for a successful channel allocation:

- Channel availability
- Carrier to Noise Ratio (CNR) between MU and BS above a predefined threshold
- Carrier to Noise plus Interference Ratio (CNIR) above a predefined threshold

Additional criteria can be applied using different channel allocation strategies. The Dynamic Channel Allocation (DCA) [32,33-36] strategy has been used in our experimental models. The CNIR ratio is derived from the following type:

$$R_{cni} = \frac{A P_0 d_0^{-\alpha} 10^{\frac{\xi_0}{10}}}{N + \sum_{i-1}^{n} A P_i d_i^{-\alpha} 10^{\frac{\xi_i}{10}}} \quad \textbf{(2)}$$

Where n is the number of base stations and users, ξi is the distortion due to shadowing from user to base station, A is a proportional coefficient, P_o is the transmitted power of a reference point, P_i is the transmitted power of the i user and d_i is the distance between MU i and reference BS.

3.2 Multi-threading scenarios

The basic network procedures such as new call admission (NC), reallocation (RC), MU movement (MC) and call termination (FC) are implemented as threads within the JVM environment and for all experimental models. Inside the simulation system, seven entities constitute the basic components. These components are:

- *Controller (clock)* which controls and synchronizes the whole simulation procedure
- *Initialization Procedures.* This procedure prepares initialization of each new simulation step (e.g. define traffic conditions, initialize counters, etc)
- *NC, RC, MC, FC* which represent the four basic network procedures respectively
- *Termination Procedures.* Actions after the completion of each simulation step (e.g. compute statistical metrics, store data for the finished simulation step, etc)

Table 1, illustrates the implemented and simulated scenarios that are based on different combinations of threads and single code tasks (S=single code, T=thread). In these scenarios four, five and seven threads have been used respectively.

	Scenario	1	2	3
		4 threads	5 threads	7 threads
1	Clock	S	T	T
2	Init loop	S	S	T
3	NC	T	T	T
4	RC	T	T	T
5	MC	T	T	T
6	FC	T	T	T
7	Termination loop	S	S	T

Table 1. Implemented scenarios

In the first scenario (fig. 3), only the basic network procedures are implemented as threads. The controller is working as a part of the main application thread and it is active until the simulation time termination. When a new simulation step starts, the controller activates the needed initializations and after the completion the four threads are activated. After completion, each thread sends a signal to the controller. When four completion signals are collected from the controller, the final loop task is activated.

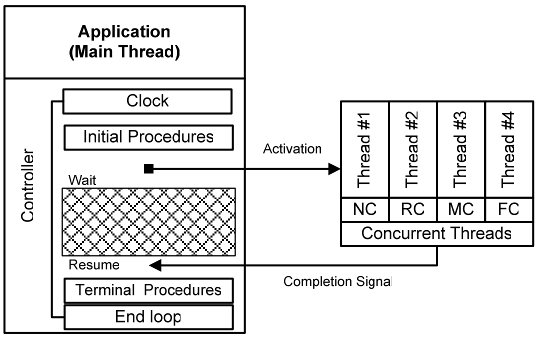

Fig. 3 The 4 thread scenario

The controller is converted to thread (fig. 4) in the case of second scenario. While the four threads are executing from JVM, the corresponding code within the application is blocked. The controller defines the activation sequence between the main simulation components. The last experimental model (fig. 5) consists of seven threads, four threads for the basic network procedures, one for synchronization and two for supplementary tasks (init loop, final loop). All the implemented threads are always active within the simulation time. Special purpose flags inside the body of each thread and in combination with control signals the corresponding code is activated. The JVM offers various methods for controlling threads. Using these methods for starting/stopping threads (instead of internal flags) significant delays and instability of the simulation models can be produced.

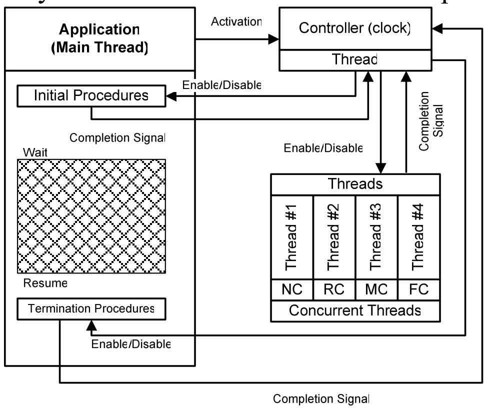

Fig. 4 The 5 thread scenario

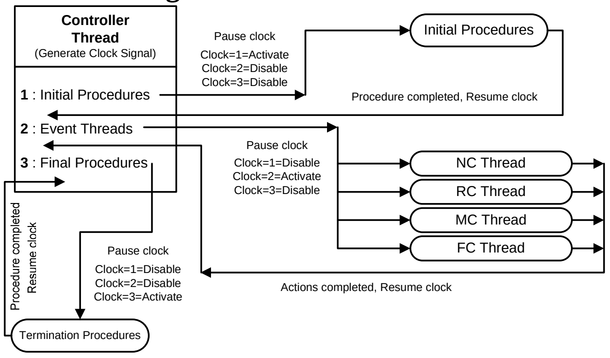

Fig. 5 The 7 thread scenario

4 DES System Evaluation

The major tasks judging performance of a cellular network are the new call admission and handoff (reallocation). This performance can be measured in terms of statistical metrics by using blocking and dropping probability [37-49]. When a new call admission is unsuccessful, then this call is blocked. The blocking probability is calculated from the type:

$$P_{blocking} = \frac{number\ of\ blocked\ calls}{number\ of\ calls} \qquad \textbf{(2)}$$

In a handoff situation, if the network can not allocate a new channel for the moving MU, then, this ongoing call is dropped. The corresponding probability is calculated as follows:

$$P_{dropping} = \frac{number\ of\ forced\ calls}{number\ of\ calls - number\ of\ blocked\ calls} \qquad \textbf{(3)}$$

For estimating more accurately the simulating results, Monte Carlo [50] executions have been used.

5 Experimental Results

Figures 6 and 7 show that by increasing thread number, network performance is improved.

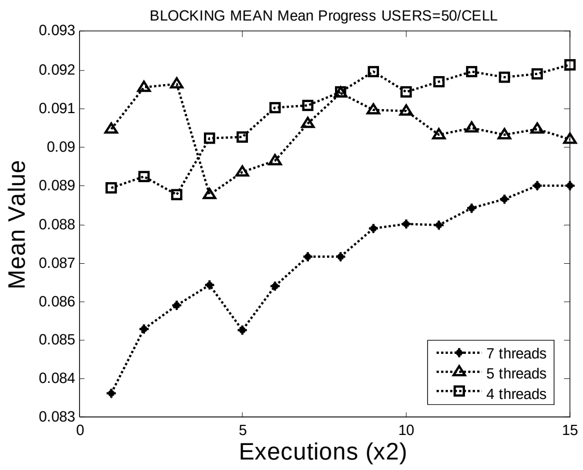

Fig. 6, Blocking probabilities for the involved architectures

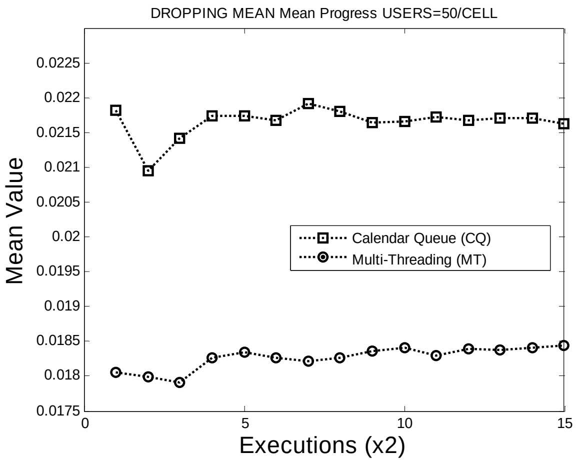

Fig. 7, Dropping probabilities for the CQ and MT approaches

Tables 2 and 3 confirm the results that are illustrated in previous graphs (figures 8,9)

Scenario	Blocking Mean Values	
	STD	MEAN
7 threads	0.044175	0.089016
5 threads	0.04337	0.090194
4 threads	0.046721	0.092127

Table 2 Mean values of STD and MEAN for blocking probability

Scenario	Dropping Mean Values	
	STD	MEAN
7 threads	0.025583	0.018442
5 threads	0.0344	0.022512
4 threads	0.033293	0.022728

Table 3 Mean values of STD and MEAN for dropping probability

Intuitively, the implemented concurrency is more realistic when the computational time for a simulation step is remaining stable (balanced between tasks inside the step). Table 4, shows the ratio *std/mean* of the simulation time duration (ms). This ratio gives us the information for the significance of the *std* of the simulation step duration and thus the resulting stability. Data inside table 4, represent results based on Monte Carlo executions. The mean values represent the ratio *std/mean* and the *std* represents the standard deviation of the resulted mean values.

	7 threads	5 threads	4 threads
Mean	0.055156	0.1479	0.20057333
Std	0.03353804	0.014321662	0.16271174

Table 4. Simulation step duration

6 Conclusions

In this paper, the influence of modeled concurrency level for the proposed simulation architectures of wireless communication systems to simulation system behavior and performance is presented. Moreover, the concept of the simulation step computational duration stability over simulation time is illustrated.

It is experimentally implied that JVM in the case of the multi-threading scenario involving implementation of a simulation architecture consisting of four threads and three simple non-thread tasks, provides balanced time slices only to the four threads. The simple non-thread tasks do not participate to this time sharing. The total computational time that is needed for each simulation step is based on the completion of each individual component such as threads and simple tasks. In the case of less than seven threads implementation the total time is asymmetrically allocated to the components and so no balanced conditions could be guaranteed. When the mentioned components are all threads, the time slicing that is provided by JVM is fairly allocated among the active threads and so balanced conditions are created. The experimental results show that there is high positive correlation between system performance and simulation time stability implying realistic reflection of the Physical Network Time.

There is no doubt that the JVM environment constitutes an effective tool for developing a multi-threading environment. On the other hand, critical drawbacks of the JVM approach (e.g. deadlocks, significant delays due to thread priorities, synchronization problems, etc) should be effectively faced by the developer. A major future research work towards developing an efficient simulation model for wireless communication systems is to focus on balancing drawbacks and benefits of such a JVM based approach.

References

[1] Reuben Pasquini, ALGORITHMS FOR IMPROVING THE PERFORMANCE OF OPTIMISTIC PARALLEL SIMULATION, PhD dissertation, Purdue University, 1999

[2] Dirk Brade, A GENERALIZED PROCESS FOR THE VERIFICATION AND VALIDATION OF MODELS AND SIMULATION RESULTS, PhD dissertation, University of Bundeswehr Munchen, 2003

[3] Rimon Barr, AN EFFICIENT, UNIFYING APPROACH TO SIMULATION USING VIRTUAL MACHINES, Cornell University, 2004

[4] R. S. M. GOH and I. L-J THNG, MLIST: AN EFFICIENT PENDING EVENT SET STRUCTURE
FOR DISCRETE EVENT SIMULATION, International Journal of SIMULATION Vol. 4 No. 5-6, 2003

[5] Gianni A. Di Caro, Analysis of simulation environments for mobile ad hoc networks, Technical Report No. IDSIA-24-03, Dalle Molle Institute for Artificial Intelligence,2003

[6] Thomas J. Schriber, Daniel T. Brunner, INSIDE DISCRETE-EVENT SIMULATION SOFTWARE: HOW IT WORKS AND WHY IT MATTERS, Proceedings of the 1997 Winter Simulation Conference, 1997

[7] Jayadev Misra, Distributed Discrete-event Simulation, ACM, Computing Surveys, Vol. 18, No. 1, March 1986

[8] Benno Jaap Overeinder, Distributed Event-driven Simulation, PhD dissertation, University of Amsterdam, 2000

[9] Bruno R. Preis, Wayne M. Loucks, V. Carl Hamacher, A UNIFIED MODELING METHODOLOGY FOR PERFORMANCE EVALUATION OF DISTRIBUTED DISCRETE EVENT SIMULATION MECHANISMS, the 1988 Winter Simulation Conference, 1988

[10] Kalyan S. Perumalla, PARALLEL AND DISTRIBUTED SIMULATION: TRADITIONAL TECHNIQUES AND RECENT ADVANCES, Proceedings of the 2006 Winter Simulation Conference,2006

[11] Mathieu Lacage, Thomas R. Henderson, Yet Another Network Simulator, ACM 2006

[12] Rick Siow Mong Goh, Ian Li- Jin Thng, DSplay: An Efficient Dynamic Priority Queue Structure For Discrete Event Simulation, ???

[13] kehsiung chung, janche sang and vernon rego, A Performance Comparison of Event Calendar Algorithms: an Empirical Approach, SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 23(10), 1107–1138 (OCTOBER 1993)

[14] Lukito Muliadi, DISCRETE EVENT MODELING IN PTOLEMY II, University of California, Berkeley, 1999

15] Valeri Naoumov, Thomas Gross Simulation of Large Ad Hoc Networks, MSWiM'03, September, San Diego, California, USA, 2003

[16] Kevin Fall, Kannan Varadhan, The ns Manual, UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 14, 2007

[17] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso, MANET Simulation Studies: The Current State and New Simulation Tools, 2005

[18] Xiang Zeng, Rajive Bagrodia, Mario Gerla, "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks", Proceedings of the 12th Workshop on Parallel and Distributed Simulations, 1998

[19] Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Rajive Bagrodia, "Simulation of Large-Scale Heterogeneous Communication Systems", Proceedings of IEEE Military Communications Conference (MILCOM'99), 1999

[20] A. Boukerche, S. K. Das, A. Fabbri, "SWiMNet: A Scalable Parallel Simulation Testbed forWireless and Mobile Networks", Wireless Networks 7, 467–486, 2001

[21] Winston Liu, Ching_Chuan Chiang, Hsiao_Kuang Wu, Vikas Jha, Mario Gerla, Rajive Bagrodia, "PARALLEL SIMULATION ENVIRONMENT FOR MOBILE WIRELESS NETWORKS", Proceedings of the 1996 Winter Simulation Conference, WSC'96, pp 605-612,1996

[22] Richard M. Fujimoto. Parallel and Distributed Simulation Systems. Parallel and Distributed Computing. Wiley-Interscience, 2000.

[23] Kevin Fall, Kannan Varadhan, (2007), The ns Manual, UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 14.

[24] Lukito Muliadi, (1999), Discrete Event Modeling In Ptolemy Ii, Department of Electrical Engineering and Computer Science University of California, Berkeley.

[25]http://jist.ece.cornell.edu/javadoc/jist/runtime/
Scheduler.Calendar.html

[26] Rick Siow Mong Goh, Ian Li- Jin Thng, (2004), DSplay: An Efficient Dynamic Priority Queue Structure For Discrete Event Simulation, SimTecT Simulation Technology and Training Conference, Australia

[27] R. S. M. GOH and I. L-J THNG, (2003), MLIST: An Efficient Pending Event Set Structure For Discrete Event Simulation, International Journal of SIMULATION Vol. 4 No. 5-6.

[28] Kehsiung Chung, Janche Sang and Vernon Rego, (1993), A Performance Comparison of Event Calendar Algorithms: an Empirical Approach, Software-Practice and Experience, vol. 23(10), 1107–1138.

[29]Oaks S., et al, "Java Threads", O'Reilly,3rd edition, 2004

[30]Kramer J.M., "Concurrency: State Models & Java Programs", John Wiley & Sons, 2nd Edition, 2006

[31]Lindsey C.S., et al, "An Introduction to Scientific and Technical Computing with java", Cambridge University Press, 2005

[32] Zhang M., and T. S. Yum, "Comparisons of Channel Assignment Strategies in Cellular Mobile Telephone Systems", IEEE Transactions on Vehicular Technology, vol.38,no.4,pp211-215, 1989

[33]Cimini L.J. and G.J. Foschini, "Distributed Algorithms for Dynamic Channel Allocation in Microcellular Systems", IEEE Vehicular Technology Conference, pp.641-644, 1992

[34]Cox D.C. and D. O. Reudink, "Increasing Channel Occupancy in Large Scale Mobile Radio Systems: Dynamic Channel Reassignment", IEEE Transanctions on Vehicular Technology, vol.VT-22, pp.218–222, 1973

[35]Del Re E., R. Fantacci, and G. Giambene, "A Dynamic Channel Allocation Technique based on Hopfield Neural Networks", IEEE Transanctions on Vehicular Technology, vol.45, no.1, pp.26–32, 1996

[36]Sivarajan K.N., R.J. McEliece, and J.W. Ketchum,"Dynamic Channel Assignment in Cellular Radio", IEEE 40th Vehicular Technology Conference, pp.631–637, 1990

[37] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey," IEEE Personal Comms., pp. 10–31, June 1996.

[38] S.H. Wong, "Channel Allocation for Broadband Fixed Wireless Access Networks", PhD dissertation, University of Cambridge, 2003

[39] P.Cherriman, F.Romiti and L.Hanzo, "Channel Allocation for Third-generation Mobile Radio Systems", ACTS'98, vol.1,pp.255-261,1998

[40] D.Grace, "Distributed Dynamic Channel Assignment for the Wireless Environment", PhD dissertation, University of York, 1998

[41] H.Haas, "Interference analysis of and dynamic channel assignment algorithms in TD–CDMA/TDD systems", PhD dissertation, University of Edinburg, 2000

[42] Hector Salgado, Marvin Sirbu, Jon Peha, "SPECTRUM SHARING THROUGH DYNAMIC CHANNEL ASSIGNMENT FOR OPEN ACCESS TO PERSONAL COMMUNICATIONS SERVICES" Proc. of IEEE Intl. Communications Conference (ICC), June 1995, pp. 417-22, 1995

[43] L.C.Godara, "Applications of Antenna Arrays to Mobile Communications, Part I: Performance Improvement, Feasibility, and System Considerations ", PROCEEDINGS OF THE IEEE, VOL. 85, NO. 7, JULY 1997

[44] Nishith D. Tripathi, Nortel Jeffrey H. Reed and Hugh F. VanLandingham, " Handoff in Cellular Systems ", IEEE Personal Communications, 1998

[45] J. Bigham, L. Du, "Cooperative Negotiation in a MultiAgent System for RealTime Load Balancing of a Mobile Cellular Network", AAMAS'03, July 14–18, 2003

[46] F. Berggren, "Power Control and Adaptive Resource Allocation in DS-CDMA Systems", PhD dissertation, Royal Institute of Technology, 2003

[47] D. Hollos, H. Karl, A. Wolisz, "Regionalizing Global Optimization Algorithms to Improve the Operation of Large Ad Hoc Networks", Proc. of IEEE Wireless Communications and Networking Conf., Atlanta, Georgia, USA, March 2004

[48] M. Cheng, Y. Li and D.-Z. Du, "Combinatorial Optimization in Communication Networks", Kluwer, 2005

[49] Wang-Chien Lee, Johnson Lee, and Karen Huff, "On Simulation Modeling of Information
Dissemination Systems in Mobile Environments ", LNCS 1748, pp. 45–57, 1999.

[50] Fishman 1995 George S, Fishman. Monte Carlo: concepts, algorithms, and applications. Springer-Verlag, (1995).