On a New Generation of Event Scheduling Algorithms and Evaluation Techniques for Efficient Simulation Modelling of Large Scale Cellular Networks Bandwidth Management Based on Multitasking Theory

P.M.PAPAZOGLOU[1,3], D.A.KARRAS[2], R.C.PAPADEMETRIOU[3]
[1] Department of Informatics & Computer Technology
Lamia Institute of Technology, Greece
p.m.papazoglou@hotmail.com, papaz@teilam.gr
[2] Department of Automation Engineering
Chalkis Institute of Technology, Greece
dakarras@teihal.gr
[3] Department of Electronic & Computer Engineering
University of Portsmouth, United Kingdom

Abstract – There are many research efforts for improving bandwidth management in wireless communication systems based mainly on DCA (Dynamic Channel Allocation) Schemes designed and evaluated through various Simulation models which, however, use a common simulation model architecture coming from queuing theory. Although much attention has been paid to Channel Allocation Mechanisms there are few only efforts related to the corresponding simulation models. These models consist of various critical components including network services models and the simulation model architecture organizing network events scheduling, network events handling and network performance evaluation. One of the most critical components is the event scheduling mechanism, which reflects network events as they happen in a real network and which has not been investigated in depth regarding its performance. The state of the art event scheduling mechanism called Calendar Queue (CQ) schedules events for later execution based on the corresponding time stamps of each generated event. The major drawback of this approach is that the generated events are executed only sequentially due to progressive time stamps. On the other hand, events in a real wireless network happen concurrently and so the state of the art mechanism can not reflect such conditions. The goal of this paper is to propose an alternative novel real time scheduling mechanism based on a synthesis of multitasking theory and queuing theory techniques, which could be involved in generating and investigating a new generation of event scheduling algorithms suitable for simulation modelling of cellular networks bandwidth management. This mechanism is analyzed through multitasking theory tools and is shown to face effectively the concurrent nature of the generated network events providing an efficient solution to the Calendar Queue problem.

Key-words: Multitasking, Network events, Scheduling, Simulation, Cellular network, Bandwidth Management

1 Introduction

1.1 Event scheduling in wireless network simulation

Simulation environments constitute efficient tools for designing and evaluating wireless cellular networks [1-6]. Discrete Event Simulation (DES) [7-16] represents the most widely known simulation methodology for wireless communication systems. The physical activities of a real wireless network are represented by events which are the main components of a DES system. Each network service constitutes an event for a particular Mobile User (MU). An event generator produces events (e.g. new voice calls) during simulation time. The scheduling mechanism constitutes a model for the event service occurrence sequence within the real wireless network. If an event is not executed, it remains in the pending event set (PES). PES, is the set of all events generated during simulation time that have not been simulated (processed) yet [10,17,18]. Thus, PES corresponds to a priority queue which controls the flow of event simulation based on current minimum time stamp (highest priority) [10]. The selected scheduling method determines how realistically the occurred real network activities will be reflected in the simulation model. In other words, scheduling is a mapping method of the real network events (activities) within the simulation time of the DES system.

1.2 The Calendar Queue (CQ) event scheduling mechanism

The major application of priority queues is the implementation of PES inside DES systems [19-26]. The mechanism based on the Calendar Queue (CQ) [23,18] concept represents the state of the art approach for event scheduling. This mechanism has been used in the most known simulation tools such as ns-2 (Berkeley, USA)[24], Ptolemy II (Berkeley, USA)[25] and Jist (Cornel University, USA)[26]. Several other variations of the CQ for improving performance of the queue itself (e.g. by optimal resizing of the queue) have been proposed in the literature such as DSplay [27], MList [10], Markov hold model [28] and SNOOPY CQ [17]. The idea of the CQ is derived from the ordinary desk calendar with one page for each day. Every event is scheduled on the appropriate page. The scheduled time of each event defines its priority. When an event is enqueued on the calendar, then this event is scheduled for future execution. The earliest event on the calendar is dequeued by searching the page for today's date and removing the earliest event written on that page [23]. Inside the computer, a CQ consists of an array of lists. Each list contains future events. According to CQ principle, the large list of *N* events is partitioned to *M* shorter lists called Buckets. Each bucket is associated with a specific range of time corresponding to future events. Any event with the occurrence time *t(e)* is associated with the *m-th* bucket in year y (y =0, 1, 2, . . .) if and only if

$$t(e) \in \left[\left((yM + m)\delta \right), \left(yM + m + 1 \right)\delta \right] \qquad (1)$$

In order to find the bucket number *m(e)* where an event *e* will occur at time *t(e)* the following type is used :

$$m(e) = \left\lfloor \frac{t(e)}{\delta} \right\rfloor \bmod M \qquad (2)$$

Assume that M=8, N=10, δ=1 and t(e)=3.40 for a new event *e*.

Using the equation (b), the bucket number for event *e* is m(e)=3.

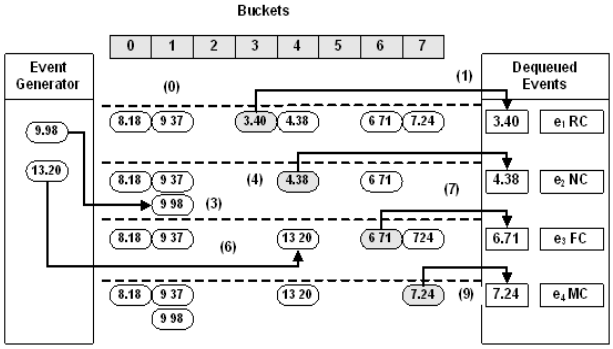


Fig. 1 Instance of a CQ with 8 buckets

Figure 1 shows a complete CQ operation for a number of generated network events. Additionally, figure 2 illustrates the execution sequence of the dequeued events inside CPU.

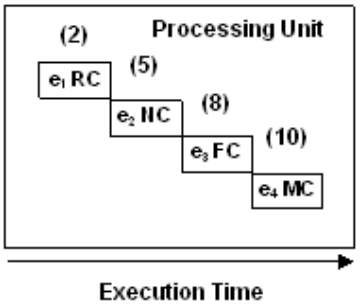


Fig. 2 Event execution sequence inside CPU

The existing state of the art event scheduling strategy (CQ) suffers in supporting concurrent network events as they happen in a real wireless network and thus an improved event scheduling mechanism is proposed.

An analytical study of the CQ can be found in [29].

1.3 Real Time Scheduling for Supporting Concurrent Tasks or Events

The real time scheduling is the most widely known mechanism for handling and executing concurrent tasks when the response time is a critical issue. Real time scheduling theory can be used for developing an alternative event scheduling as compared to the state of the art scheduling mechanism found in the DES systems to support also concurrent events.

Real-time systems (RTSs), are widely used in a number of critical applications such as robotics, avionics, telecommunications, process control. RTSs are also used in numerous embedded systems. The common feature of the above systems is the restricted deadlines for the results' production. In other words, the timing of the results is critical and even ruinous in some cases. According to [30] "***Real-time systems are those in which the correctness of the system depends not only on the logical results of computation, but also on the time at which the results are produced***". According to [31], various components are required for the interaction between the real-time control system and the environment such as sensors that read the state of the external environment. Software architectures constitute common implementation of RTSs. Based on [31,32], these architectures can be classified in (a) cyclic executives, (b) concurrent tasks systems activated by events, (c) message passing systems and

(d) client-server systems. In concurrent tasks systems [31], a specialized scheduler is needed in order to take scheduling decisions due to the fact that the notion of a task is maintain at real-time. A way to predict the timing behaviour of complex Multi Tasking (MTAS) software is the real-time scheduling theory. Since the first formulation of the scheduling theory, several improvements have been introduced such as Rate Monotonic Analysis in [33] and Deadline Monotonic Analysis in [34]. These two improvements have been integrated in the fixed-priority scheduling theory [35]. Some very important characteristics of the RTSs found in [36], are (a) repeated interaction with the environment, (b) perform multiple actions at the same time, switch rapidly to events and involve high degree of concurrency, (c) competition for shared resources, (d) actions triggered externally by events (within the environment) or after time progress, (e) stability in overloads and (f) maintainability and extensibility.

1.3.1 Task lifecycle in a real-time scheduling environment

The task lifecycle in a RTS can be analyzed by using the single queue approach combined with the pre-emption mechanism.

A RTS, consists of the following three basic components [37] (a) the set which contains the computational tasks to be performed (typical subroutines with private thread of control), (b) a run-time scheduler (e.g. dispatcher) which defines the task execution sequence, (c) the set with shared resources that tasks access. Figure 3 illustrates the typical lifecycle of task execution from occurrence to termination through a queue.

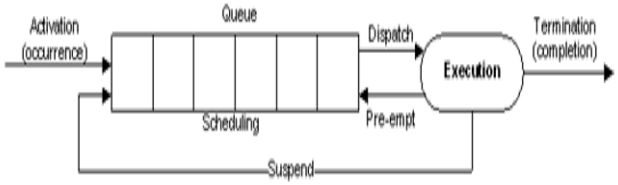


Fig. 3 Task lifecycle

The tasks that are served by the system can be categorized by the time characteristics of arrival as (a) Periodic (fixed arrival intervals) [36], (b) Aperiodic (random arrivals) [38] and (c) Sporadic (with only a fixed minimum interarrival time) [38].

1.3.2 Scheduling algorithms categorization

The scheduling algorithms can be categorized in general as static or dynamic [36]. In static scheduling, the schedule is defined before any task execution. On the other hand, in dynamic scheduling the decisions about which task to execute are taken at run-time. For multiple tasks, a priority must be defined in order to know which task will be pre-empted by another and what to do when more than one task is active at the same time. The task priorities can be fixed or dynamic based on current needs. When fixed priorities are applied, the task execution sequence does not change during execution in contrast to dynamic priorities that can be changed during run-time period.

1.3.3 Pre-emptive fixed priority scheduling (PFP)

In PFP scheduling, each task has a fixed priority that does not change during run-time. The main advantages of this approach are (a) faster service of tasks with higher priority [37,39], (b) simpler implementation, (c) stability in overloads.

Figure 4 shows four tasks with different priorities (Low1<Low2<High1<High2) during specific time widow of execution.

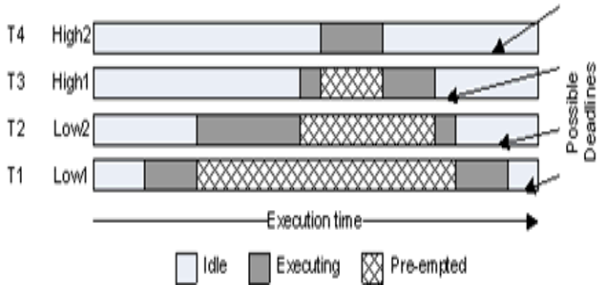


Fig. 4 Pre-emptive fixed priority scheduling (Tasks/Events: T1,T2,T3,T4)

For example, while T1 event is under processing, a new event (T2) has arrived (fig. 4).

1.3.4 Pre-emptive dynamic priority scheduling (PDP)

When the PDP scheduling is used, the highest priority is given to the task with the earliest deadline. This scheduling method serves the first tasks with the most important needs and so there is a guarantee for the required deadlines. On the other hand, it is not stable under peak overloads due to the fact that is difficult to predict accurately which deadlines will be missed first.

1.4 Response Time Analysis

The whole system can be viewed as a real-time MTAS system due to task occurrence while other tasks are under processing. This approach leads to a more advanced event control and interleaving. Based on MTAS concepts and schedulability analysis [37], events behaviour is analyzed as follows:

Any event e, with $e \in \{NC, RC, MC, FC\}$ has a release time r:

$$r_e = (n \cdot T_e) - J_e \quad (3)$$

Where n is the n-th event, T_e is the event period and J_e is the corresponding Jitter. The J_e is calculated as

$$J_e = r_e(max) - r_e(min) \quad (4)$$

In the calculations it is assumed that $J_e=0$.
Computational time for an event is

$$C = (n+1) \cdot C_e \quad (5)$$

Let R_i, the response time for task i, C_i the worst-case computational time and I_i the interference (due to higher priority tasks). The response time for task i, is:

$$R_i = C_i + I_i \quad (6)$$

Assuming that all priorities are unique, the interference for task i, is:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (7)$$

where hp(i) is the set of all tasks that have greater priority than i.
from equations (6) and (7), the worst-case response time of task I can be calculated as follows:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (8)$$

Equation (8) can be solved iteratively [40], starting with the initial approximation for R_i of 0 (invocation). Knowing the x-th, the (x+1) can be approximated as follows:

$$R_i^{x+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^x}{T_j} \right\rceil C_j \quad (9)$$

The main and useful conclusion that stems from equation (9) is the schedulability of task i, which means that the required deadlines can be met. For safe conclusions, the above equation must be evaluated repeatedly in order to see if it converges to a value R_i such that $R_i<D_i$ (D_i=the given deadline for task i). Instead the existence of specialized tools for evaluating the above equations, such tools do not give necessarily any insight into the dynamic behaviour of each task to the programmer [37]. Simulation is the most appropriate approach to display the time behaviour of the given tasks [37].
In order to perform complete calculations the event priorities must be set.

2 The proposed event scheduling mechanism

2.1 Supported Network Services-Events

The developed and evaluated network simulation model supports four different network services to candidate and active MUs. These services are:

- New call admission (NC). The cellular network assigns one or more communication channels to a new MU.
- Channel reallocation (RC). When the signal quality is below the Carrier to Noise plus Interference Ratio (CNIR) threshold, the network tries to assign new acceptable channels.
- User movement (MC). MU movement based on a daily traffic profile. The network assigns also new channels based on the new MU location.
- Call termination (FC). The call is terminated when the call holding time is expired.

2.2 The Multiple Queue Model

In a wireless environment, where events happen concurrently, the performance of the network depends not only on the logical results of computation, but also on the time sequence at which the results are produced. In other words, it depends on the logical sequence of the various network procedures regarding the efficient bandwidth management. Network events such as new call admission, reallocation, call termination, etc, can be faced as tasks that have to be served by the network. The scheduling mechanism that manages task serving plays a major role in the resulting network performance. In other words, this mechanism defines how the network will serve the concurrent tasks (events) in the most efficient way in terms of network performance metrics. Applying response time analysis as a first step, the event service progress and behaviour is shown. Due to one processor existence in conventional computers, event execution is sequential. The most known simulation tools, such as ns-2, are based on CQ type scheduling which represents the sequential logic. In this approach the execution sequence is defined from the time stamps of each generated event. CQ holds event information for future execution. These time stamps can be viewed also as priorities. When the network events are faced as concurrent, the Multi Tasking (MTAS) conceptual approach extended properly can be applied. In a concurrent model, when a MU is under processing from the network, another MU is

moving or trying for reallocation. Thus, the task (event) execution has to be partial, in order to handle the other concurrent MUs. This logic leads to the Priority Queue-Time Division Multiplexing (PQ-TDM) that usually is implemented by using Multi Threading (MT) technologies.

Figure 5 illustrates the queuing models for each approach (sequential & concurrent).

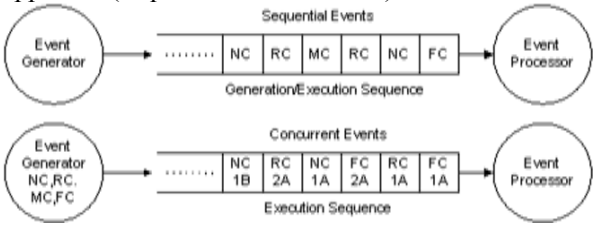


Fig. 5 Queuing models for sequential and concurrent events

In a concurrent environment, events can be faced as multiple tasks that must be served by the available processor (fig. 6).

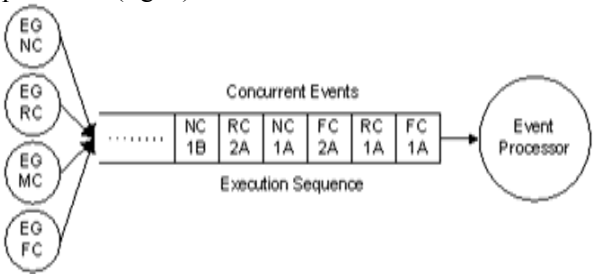


Fig. 6 Multiple Event Generators (EGs), one server (Event processor)

In order to control the concurrent event partial execution sequence inside the PQ-TDM queue, four more queues are used for representing the corresponding network events. Figure 7 shows the multiple queue system which supports all the MU requests. Events of the same type are stored in a common queue (NCQ through FCQ). The PQ-TDM mechanism selects the appropriate event parts based on the defined priorities and the corresponding execution time slot. Finally, the main PQ-TDM queue contains the PES list.

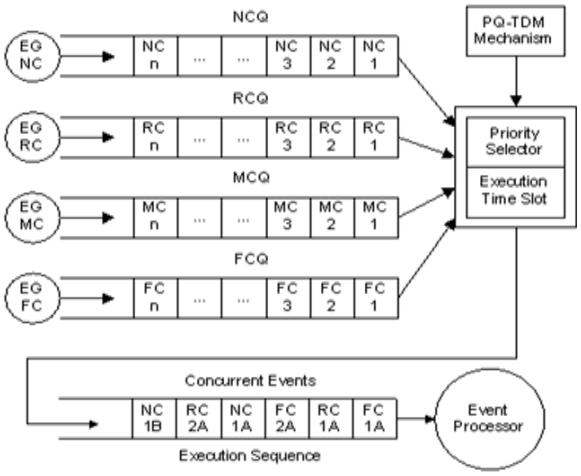


Fig. 7 The multiple queue system

2.3 Response Time Analysis Adapted to Network Events

Let P_{FC}, P_{RC}, P_{MC}, P_{NC} the priorities of the event types FC, RC, MC and NC respectively with $P_{FC}>P_{RC}$, $P_{RC}>P_{MC}$, $P_{MC}>P_{NC}$. Based on the above definitions, the corresponding hp(j) sets are:

$$hp(FC)=\{\varnothing\} \tag{10}$$

$$hp(RC)=\{FC\} \tag{11}$$

$$hp(MC)=\{RC,FC\} \tag{12}$$

$$hp(NC)=\{MC,RC,FC\} \tag{13}$$

Equation (9) can be solved iteratively [40]. Thus, the response time for next event n+1 can be calculated as:

$$R_e^{n+1}=C+\sum_{j\in hp(e)}\left\lceil\frac{R_e^n}{T_j}\right\rceil\cdot C_j \tag{14}$$

Formulating the above equations to network services, the resulting response times are:

Assuming that $R_e^0=0$,

$$R_{FC}^{n+1}=C_{FC}+\sum_{j\in hp(FC)}\left\lceil\frac{R_{FC}^n}{T_j}\right\rceil\cdot C_j \Rightarrow R_{FC}=C_{FC} \tag{15}$$

since $hp(FC)=\{\varnothing\}$

$$R_{RC}^{n+1}=C_{RC}+\left\lceil\frac{R_{RC}^n}{T_{FC}}\right\rceil\cdot C_{FC} \tag{16}$$

$$R_{MC}^{n+1} = C_{MC} + \left\lceil \frac{R_{MC}^{n}}{T_{RC}} \right\rceil \cdot C_{RC} + \left\lceil \frac{R_{MC}^{n}}{T_{FC}} \right\rceil \cdot C_{FC} \quad (17)$$

$$R_{NC}^{n+1} = C_{NC} + \left\lceil \frac{R_{NC}^{n}}{T_{MC}} \right\rceil \cdot C_{MC} + \left\lceil \frac{R_{NC}^{n}}{T_{RC}} \right\rceil \cdot C_{RC} + \left\lceil \frac{R_{NC}^{n}}{T_{FC}} \right\rceil \cdot C_{FC} \quad (18)$$

Based on the above rules, a pre-emptive event scheduling can be applied.
The lower priority events (tasks) are pre-empted from a higher priority event.
Pre-emptive fixed priority scheduling (PFP) and pre-emptive dynamic priority scheduling (PDP) scheduling strategies have been adapted to the final proposed simulation methodology. As previously mentioned, when a MU is under servicing, simulation time flows. Within servicing time, another MU call may arrive. Let each simulation step with duration N sec (e.g. 1 sec) and let m_iN the required computational time (RCT) for each type i of event service. Thus, each service is completed at time t+m_iN (simulation time t+N, t+2N, …, t+m_iN). Figure 8 shows an example where the network events arrive at different points in time.

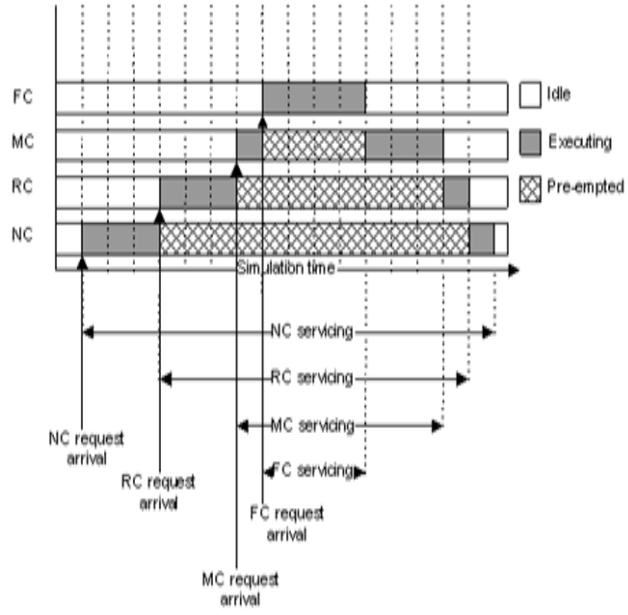


Fig. 8 Servicing multiple events based on pre-emptive scheduling (m=4, N=1sec)
(Tasks/Events: NC-New Call, RC-Call Reallocation, MC-Movement Call, FC-Finished Call)

The total response time for each queue type can be calculated by adding the partial response times of the enqueued events. If n is number of the total events inside an event queue, then the total response time for each type of service can be calculated as follows:

$$Total\ R_{FC} = \sum_{i=1}^{n} C_{FC} = nC_{FC} \quad (19)$$

$$Total\ R_{RC} = \sum_{i=1}^{n}\left(C_{RC} + \left\lceil \frac{R_{RC}^{i-1}}{T_{FC}} \right\rceil \cdot C_{FC} \right) = \\ = \sum_{i=1}^{n} C_{RC} + C_{FC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{RC}^{i-1}}{T_{FC}} \right\rceil \right) = nC_{RC} + C_{FC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{RC}^{i-1}}{T_{FC}} \right\rceil \right) \quad (20)$$

$$Total\ R_{MC} = \sum_{i=1}^{n}\left(C_{MC} + \left\lceil \frac{R_{MC}^{i-1}}{T_{RC}} \right\rceil \cdot C_{RC} + \left\lceil \frac{R_{MC}^{i-1}}{T_{FC}} \right\rceil \cdot C_{FC} \right) = \\ = \sum_{i=1}^{n} C_{MC} + C_{RC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{MC}^{i-1}}{T_{RC}} \right\rceil \right) + C_{FC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{MC}^{i-1}}{T_{FC}} \right\rceil \right) = \\ = nC_{MC} + C_{RC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{MC}^{i-1}}{T_{RC}} \right\rceil \right) + C_{FC} \sum_{i=1}^{n}\left(\left\lceil \frac{R_{MC}^{i-1}}{T_{FC}} \right\rceil \right) \quad (21)$$

$$Total R_{NC} = \sum_{i=1}^{n}\left(C_{NC} + \left\lceil \frac{R_{NC}^{i-1}}{T_{MC}} \right\rceil \cdot C_{MC} + \left\lceil \frac{R_{NC}^{i-1}}{T_{RC}} \right\rceil \cdot C_{RC} + \left\lceil \frac{R_{NC}^{i-1}}{T_{FC}} \right\rceil \cdot C_{FC} \right) = \\ = nC_{NC} + C_{MC} \sum_{i=1}^{n}\left\lceil \frac{R_{NC}^{i-1}}{T_{MC}} \right\rceil + C_{RC} \sum_{i=1}^{n}\left\lceil \frac{R_{NC}^{i-1}}{T_{RC}} \right\rceil + C_{FC} \sum_{i=1}^{n}\left\lceil \frac{R_{NC}^{i-1}}{T_{FC}} \right\rceil \quad (22)$$

For theoretical evaluation, let the following event and scheduling characteristics:

- Poisson event arrival times with λ=10
- Number of events=15 (executions)
- Computational event time=2 time steps
- Priorities $P_{NC}<P_{MC}<P_{RC}<P_{FC}$

Table 1 shows the partial and total response times based on the event times using the equations (15) to (22).

Event	Response times	Total
FC	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2	30
RC	2,4,4,4,4,4,4,4,4,4,4,4,4,4,4	58
MC	2,6,6,8,10,10,10,10,8, 8,8,8,8,8,10	120
NC	2,8,10,12,18,20,16,16, 14,14,16,16,14,14,16	206

Table 1. Partial and total Response times

2.4 The derived Proposed Priority Queue (PQ) –TDM event scheduling mechanism

The proposed PQ-TDM Layered Multithreading algorithm extends the CQ paradigm to handle concurrent events in simulation models of complex systems, where event occurrence time cannot be specified as easily as in the CQ described simulation

systems. The basic entity of the algorithm, the event, is specified as a thread. According to the PQ-TDM principle, the list of N concurrent/non-concurrent events taking place within the system is partitioned to shorter lists called Priority Buckets. Moreover, there exists a basic periodic event-thread, the Time Clock, which synchronizes all other events-threads. This event-thread is associated with the largest Priority PMAX. Provided that the Multithreading Simulation Platform supports P priorities, let's DP be the priority distance between PMAX priority of the Time Clock event-thread and the priority associated with the priority bucket having the largest priority. If we assume that the priorities supported are 1,2,…P, then PMAX= P, and PMAX - DP are the supported buckets in which the event list is partitioned. Each such bucket, on the other hand, is associated with a specific range of priorities corresponding to future events. Any event with the occurrence priority p(e) is associated with the m-th bucket in Basic Priority p (p =0,1,2, ..) if and only if

$$p(e) \in [(p(P_{MAX}-D_P)+m)\delta, (p(P_{MAX}-D_P)+m+1)\delta] \quad (19)$$

In order to find the bucket number m(e) where an event e belongs with priority p(e) the following formula is introduced:

$$m(e) = \left| \frac{p(e)}{\delta} \right| \bmod (P_{MAX}-D_P) \quad (20)$$

Regarding time t(e) of the event-thread e, it should be remarked that now it is determined by the Multithreading Simulation Platform by a Time Division Multiplexing (TDM) procedure. Time slice ΔT is the basic entity in this TDM procedure. That is, ΔT computational time is given to each event out of the events list to proceed its computations, within which it might finish or not. If it doesn't finish then, it waits for a future assignment of a ΔT computational time again. Events with higher priority are assigned with more such ΔT time slices. An important aspect in the proposed scheduling algorithm is that DP should be as maximum as possible in order for the Time-Clock Thread to be a reliable controller of the multithreading architecture of events-threads and face the known difficulties of multithreading technology to reliably schedule threads, due to absence of such specifications and definitions in Multithreading Simulation Platforms like JVM etc.

3 Scheduling Mechanism and Network Model validation

3.1 Scheduling mechanism validation

Initially, a reference model has been built in a conceptual level in order to verify the correctness of the PQ-TDM approach compared to existing CQ approach. Due to the simplicity of the conceptual model, one only cell is assumed for serving MUs within the coverage area. Figure 9, illustrates that cell that offers n channels for supporting new call admission, reallocation and MU movement. This model does not involve any advanced mathematical model (e.g. for signal propagation) because it is focused on the way that the MUs are served by the channels.

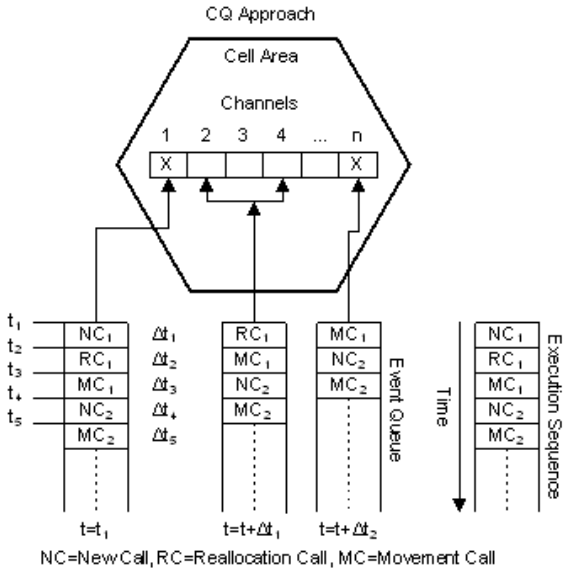


Fig. 9 MU service operation based on CQ approach

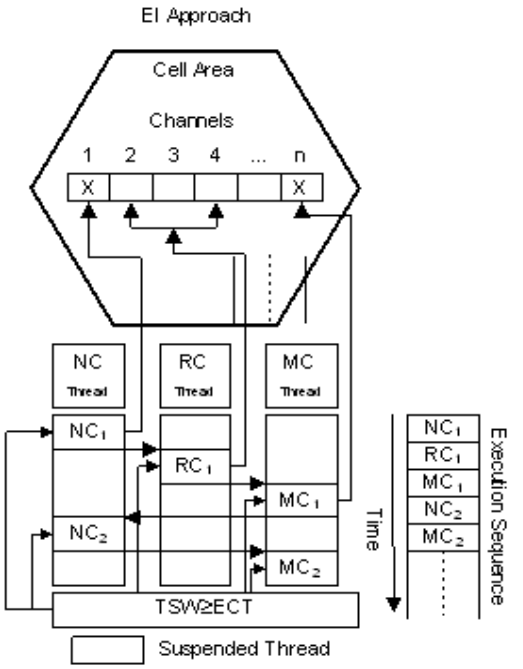


Fig. 10 MU service operation based on PQ-TDM (EI- Event Interleaving approach)

Both models have been tested initially with deterministic data for event generation. Figure 9, shows how some of the five (NC_1, NC_2, RC_1, MC_1, RC_2) events are executed. The scheduler restores from the queue the event with the lower time stamp (highest priority) for the next execution. When the same events are also generated in the PQ-TDM system (fig. 10), the execution sequence remains the same while the given time slice (Time Slice Width - TSW) from the scheduler is greater or equal to Event Computational Time (ECT). In other words, if TSW>ECT then one time slice is enough for the completion of each generated event in the predefined sequence. Thus, the two conceptual models produce the same results. When the TSW is less than the ECT for event completion or the TSWs are asymmetrically assigned to active threads, the results are totally different from the conventional CQ approach. This happen because of the competition between running threads (e.g. individual MUs) for common resources (e.g. radio channels) and the final channel allocation achievement in different points in time.

Let the event execution with the following characteristics:

If NCp=3, RCp=1, MCp=1, FCp=1, PMAX=10, MAEP=3 and N=3, then the resulted buckets are as follows (fig. 11):

Pri	**B. No**					**Multi**
3	0	NC_n	…	NC_2	NC_1	x 3
2	1	Not used				x 2
1	2	n	…	RC_1	FC_1	x 1

Fig. 11 Bucket structure (Pri=Priority, B.No=Bucket number, Multi=Time slice multiplier)

According to figure 11, the execution starts with the first NC event (NC_1) for time TSW x 3 (TSW=Time Slice Width) and after that time the event FC_1 is executed next for TSW x 1 time, and so on. Figure 12 shows the execution interleaving according to the example of figure 11.

	Execution Sequence (interleaving)					
Time	x 3	x 1	x 1	x 3	x 1	x 1
Event	NC_1	FC_1	RC_1	NC_2	FC_1	RC_1

Fig. 12 Sample of event execution

3.2 Network Model Validation

The implemented algorithms and models in this paper have been tested in a simulation environment that integrates the basic simulation and network components. The necessary validation of the simulation environment consists of three different validation levels which are:

- Calendar Queue (State of the art) scheduling mechanism implementation algorithm
- Network environment which includes signal propagation, interference and signal measurements
- Network performance compared to theoretical computations

The most known statistical metrics for the wireless network performance evaluation are blocking and dropping probabilities [41-49]. Blocking probability represents the blocked calls, while dropping represents the unsuccessful channel reallocation for an ongoing call. The dropping probability is strongly connected to Carrier to Noise plus Interference Ratio (CNIR), because when this ratio is not above the accepted threshold and the network can not allocate an appropriate channel, the call is dropped.

Figure 13 shows the theoretical blocking probability as compared to the simulated blocking probability, derived in the proposed wireless communication system simulation system. The simulated probability has been generated from the large scale network based on the classical DCA algorithm [49] and the above mentioned network services.

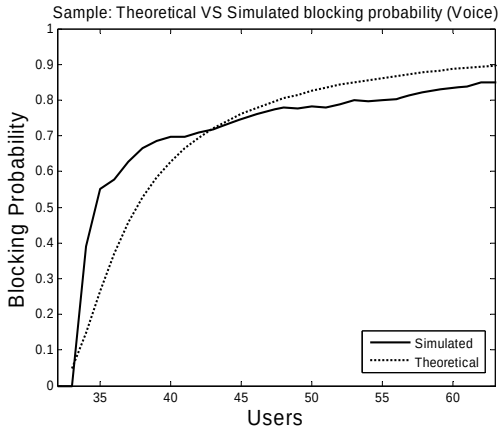


Fig. 13 Theoretical blocking probability versus simulated.

Other measures too, like the ones used in [50] based on delays measurements, are under examination to be involved in the present framework by the authors, since the classical blocking and dropping probabilities estimation, suitable for voice services,

can not accommodate all needed evaluations for state of the art cellular systems providing many additional multimedia services.

4 Simulation results

4.1 Network performance

Figure 14 shows the performance of the proposed event scheduling mechanism as compared to the state of the art approach in terms dropping probability.

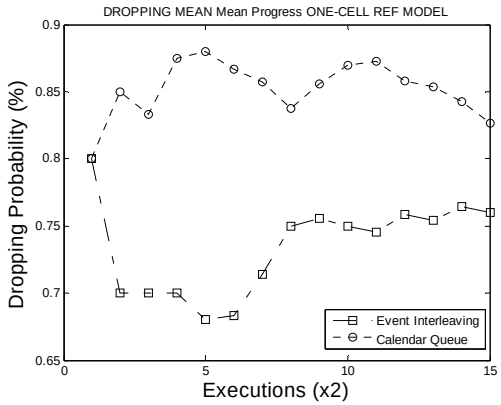


Fig. 14 Dropping probability of CQ and PQ-TDM (Event Interleaving) approaches (Monte Carlo executions, one-cell reference model)

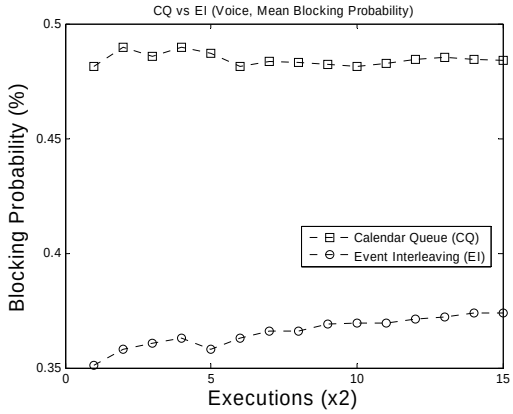
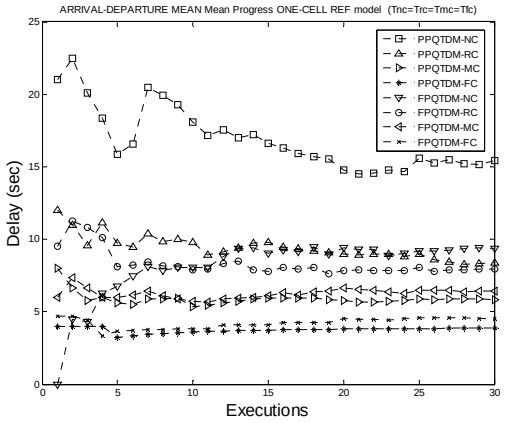


Fig. 15 State of the art CQ implemented in a conventional platform Versus Proposed PQ-TDM (Event Interleaving) implemented in multi threading platform (Voice, Mean Blocking Probability, Classical DCA, Monte Carlo executions)

Figure 15, represents the performance of the proposed event scheduling mechanism as compared to the state of the art approach in terms of blocking probability. These results represent a simulated large scale cellular network.

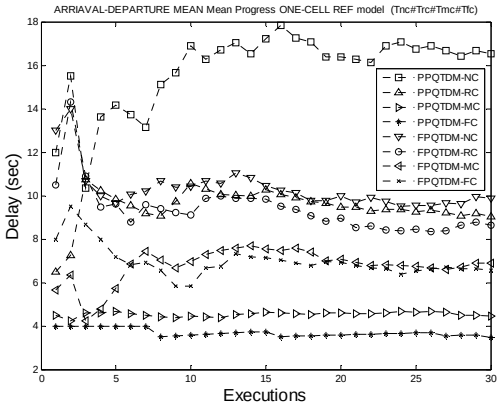
4.2 Scheduling mechanism performance

Figure 16 shows the service time (response time) for each requested network service (event) process based on the one-cell reference model. The same figure shows also the stability of the model. The graph points in figure 16 represent two variations of the PQ-TDM mechanism. According to Pre-emptive PQ-TDM (PPQTDM) the given execution time is based on the defined priorities of the arrived events. On the other hand, when the event arrival periods are comparable or less than the time (represented by the event computational times) that the network needs to complete the services, the events with lower priority can not be served. This problem is solved by the (Fair) FPQTDM which gives equal execution time to the arrived events and the execution order is based on the defined priorities. Figure 16 shows response time for equal arrival times and figure 17 represents response time results for non equal event arrival times.



PPQTDM-NC = Pre-emptive PQ-TDM scheduling for New Call Admission
PPQTDM-RC = Pre-emptive PQ-TDM scheduling for Reallocation Call (Handoff)
PPQTDM-MC = Pre-emptive PQ-TDM scheduling for Movement Call (MU movement)
PPQTDM-FC = Pre-emptive PQ-TDM scheduling for Finished Call (Call Termination)
(Fair) FPQTDM-NC = Equal computational period for New Call Admission
(Fair) FPQTDM-RC = Equal computational period for Reallocation Call (Handoff)
(Fair) FPQTDM-MC = Equal computational period for Movement Call (MU movement)
(Fair) FPQTDM-FC = Equal computational period for Finished Call (Call Termination)

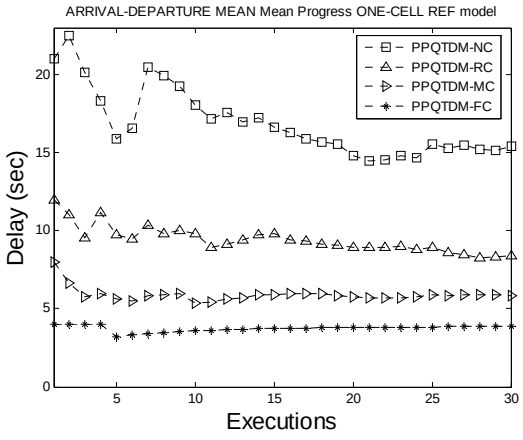
Fig. 16 Delay measurements between event arrival and departure times (equal arrival times)



PPQTDM-NC = Pre-emptive PQ-TDM scheduling for New Call Admission
PPQTDM-RC = Pre-emptive PQ-TDM scheduling for Reallocation Call (Handoff)
PPQTDM-MC = Pre-emptive PQ-TDM scheduling for Movement Call (MU movement)
PPQTDM-FC = Pre-emptive PQ-TDM scheduling for Finished Call (Call Termination)
(Fair) FPQTDM-NC = Equal computational period for New Call Admission
(Fair) FPQTDM-RC = Equal computational period for Reallocation Call (Handoff)
(Fair) FPQTDM-MC = Equal computational period for Movement Call (MU movement)
(Fair) FPQTDM-FC = Equal computational period for Finished Call (Call Termination)

Fig. 17 Delay measurements between event arrival and departure times (non equal arrival times)

Figure 18 shows clearly the response time for the four supported network services.



PPQTDM-NC = Pre-emptive PQ-TDM scheduling for New Call Admission
PPQTDM-RC = Pre-emptive PQ-TDM scheduling for Reallocation Call (Handoff)
PPQTDM-MC = Pre-emptive PQ-TDM scheduling for Movement Call (MU movement)
PPQTDM-FC = Pre-emptive PQ-TDM scheduling for Finished Call (Call Termination)

Fig. 18 Response time based on the PPQTDM

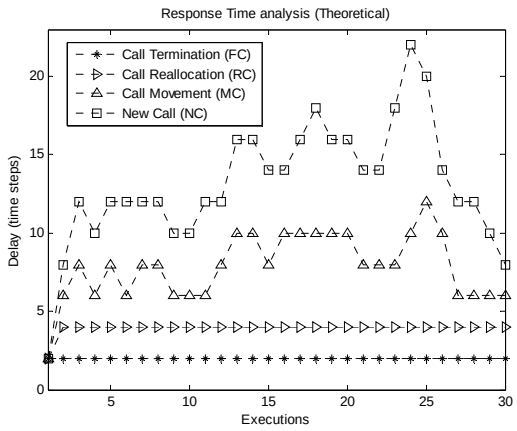


Fig. 19 Theoretical response time (non equal arrival times)

The response time of the tested model can be evaluated using the theoretical results that are based on the same mechanism.

5 Conclusions and Future Work

An efficient real time scheduling mechanism for concurrent network events is proposed in this paper based on merging queuing and multitasking theory techniques. It is shown how a new generation of event scheduling algorithms for simulation modelling of large scale cellular networks bandwidth management could be realized by involving this mechanism. The key concept of the proposed mechanism is the time division multiplexing of the competitive network events for the available network resources such as the channels. Moreover, it has been illustrated how multitasking theory techniques, like response time analysis could provide the means for evaluating such algorithms. This new framework is applied to the bandwidth management problem of cellular networks and it is shown that the proposed simulation model and analysis is in the right direction for investigating and improving simulation modelling in cellular systems.

References

[1] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou "An Efficient Scheduling Mechanism for Simulating Concurrent Events in Wireless Communications Based on an Improved Priority Queue (PQ) TDM Layered Multi-Threading Approach", WSEAS Transactions on Communications, Issue 3, vol 7, 2008

[2]P.M.Papazoglou, D.A.Karras, R.C.Papademetriou "High Performance Novel Hybrid DCA algorithms for efficient Channel Allocation in Cellular Communications modeled and evaluated through a Java Simulation System", WSEAS Transactions on Communications, ISSN 1109-2742,Issue 11, vol 5, 2006, pp.2078-2085

[3] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou "On new dynamic channel assignment schemes and their efficient evaluation through

a generic simulation system for large scale cellular telecommunications", HERMIS, An International Journal of Computer Mathematics and its Applications, Vol. 6, 2006.

[4] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "Novel DCA algorithms for efficient Channel Assignment in Cellular Communications and their evaluation through a generic Java Simulation System", 6th WSEAS Int. Conf. on SIMULATION, MODELLING AND OPTIMIZATION (SMO '2006)

[5] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "Evaluating new efficient DCA schemes through a Generic Simulation System for large scale GSM cellular telecommunications", The 15th IASTED International Conference on APPLIED SIMULATION AND MODELLING, June 26-28, 2006, Rhodes, Greece

[6] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "Concurrent Events Scheduling for Efficient Simulation Modelling of Large Scale Cellular Networks Based on Multitasking Real Time Scheduling Techniques and Analysis", 12th WSEAS Int. Conf. on COMMUNICATIONS, 2008

[7] Pasquini, R. (1999). Algorithms for improving the performance of optimistic parallel simulation. Unpublished Doctoral dissertation, Purdue University.

[8] Brade, D. (2003). A generalized process for the verification and validation of models and simulation results. Unpublished Doctoral dissertation, University of Bundeswehr, Munchen.

[9] Barr, R. (2004). An efficient, unifying approach to simulation using virtual machines. Cornell University.

[10] Goh, R. S. M., & Thng, I. L.(2003). MLIST: An efficient pending event set structure for discrete event simulation, international journal of simulation. 4(5-6).

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

[12] Schriber, T.J., & Brunner, D.T. (1997). Inside discrete-event simulation software: how it works and why it matters. Proceedings of the 1997 Winter Simulation Conference.

[13] Misra, J. (1986). Distributed Discrete-event Simulation. ACM Computing Surveys, 18(1).

[14] Overeinder, B.J. (2000). Distributed Event-driven Simulation. Unpublished Doctoral dissertation, University of Amsterdam.

[15] Preiss, B.R., Loucks, W.M. & Hamacher, V.C. (1988). A unified modeling methodology for performance evaluation of distributed discrete event simulation mechanisms. The 1988 Winter Simulation Conference.

[16] Perumalla, K.S. (2006). Parallel and distributed simulation: traditional techniques and recent advances. Proceedings of the 2006 Winter Simulation Conference.

[17] Tan, K.L., & Thng, L.J. (2000). Snoopy Calendar Queue. Proceedings of the 2000 Winter Simulation Conference.

[18] Siangsukone, T., Aswakul, C., & Wuttisittikulkij, L. (2003). Study of Optimised bucket widths in Calendar Queue for Discrete Event Simulator. Thailand's Electrical Engineering Conference (EECON-26).

[19] Blackstone, J.H., Hogg, C.L., & Phillips, D.T. (1981). A two-list synchronization procedure for discrete event simulation. Commun ACM, 24(12), 825-629.

[20] Henriksen, J.O. (1977). An improved events list algorithm. Proceedings of the 1977 Winfer Simulation Conference (Gaithersburg. Md., Dec. 5-7). IEEE, Piscataway, N.J. pp. 547-557.

[21] Kingston, J. H. (1984). Analysis of Tree algorithms for the simulation event list. Unpublished Doctoral dissertation, Basser Dept. Computer Science, University of Sydney, Australia.

[22] McCormack, W.M., & Sargent, R.G. (1981). Analysis of future event-set algorithms for discrete event simulation. Communications of the ACM, 24(12), 801-812.

[23] Brown, R. (1988). Calendar queues: A fast O (1) priority queue implementation for the simulation event set problem. Communications of the ACM, 31(10), 1220–1227.

[24] Fall, K., & Varadhan, K. (2007). The ns Manual. UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 14.

[25] Muliadi, L. (1999). Discrete event modeling in ptolemy ii. University of California, Berkeley.

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

[27] Siow, R., Goh, M., & Thng, I.L.J. (2004). DSplay: An Efficient Dynamic Priority Queue Structure For Discrete Event Simulation. Simtect Simulation Conference.

[28] Chung, K., Sang, J., & Rego, V.A. (1993). Performance Comparison of Event Calendar Algorithms: an Empirical Approach. Software—Practice and Experience, 23(10), 1107–1138.

[29] Erickson, K.B., Ladner, R.E., & LaMarca, A. (1994). Optimizing Static Calendar Queues. Annual IEEE Symposium on Foundations of Computer Science, 35, 732-743.

[30] Burns, A., & Wellings, A. (1996). Real-Time Systems and Programming Languages (2nd ed.). Addison-Wesley.

[31] Koch, B. (1999). The theory of task scheduling in real-time systems. Student Thesis, University of Hamburg.

[32] Locke, C. D. (1992). Software architecture for hard real-time applications: Cyclic executives vs. fixed priority executives. The Journal of Real-Time Systems, 4, 37–53.

[33] Liu, C. L., & Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1), 46–61.

[34] Audsley, N. C., Burns, A., Richardson, M. F., & Wellings, A. J. (1992). Deadline monotonic scheduling theory. Proceedings of the 18th IFAC/IFIP Workshop on Real-Time Programming (WRTP'92).

[35] Audsley, N.C., Burns, A., Davis, R. I., Tindell, K. W., & Wellings, A. J. (1995). Fixed priority pre-emptive scheduling: An historical perspective. Real-Time Systems, 8, 173–198.

[36] Buttazzo, G.C. (1997). Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and applications. Kluwer Academic Publishers.

[37] Fidge, C.J. (2002). Real-time scheduling theory. Technical report. University of Queensland.

[38] Sprunt, B. (1990). Aperiodic Task Scheduling for Real-Time Systems. Unpublished doctorate dissertation, Carnegie Mellon University.

[39] Kalinsky, D. (2001). Context switch, Embedded Systems Programming

[40] Audsley, N., Burns, A., Richardson, M., Tindell, K., & Wellings, A. (1993). Applying new scheduling theory to static priority pre-emptive scheduling. Software Engineering Journal, 8(5), 284–292.

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

[42] Cheng, M., Li, Y., & Du, D.Z. (2005). Combinatorial Optimization in Communication Networks. Kluwer Academic Publishers.

[43] Cherriman, P., Romiti, F., & Hanzo, L. (1998). Channel Allocation for Third-generation Mobile Radio Systems. ACTS' 98, 1, 255-261.

[44] Godara, L.C. (1997). Applications of Antenna Arrays to Mobile Communications, Part I: Performance Improvement, Feasibility, and System Considerations. Proceedings of the IEEE, 85(7).

[45] Grace, D. (1998). Distributed Dynamic Channel Assignment for the Wireless Environment. Unpublished Doctoral dissertation, University of York.

[46] Haas, H. (2000). Interference analysis of and dynamic channel assignment algorithms in TD–CDMA/TDD systems. Unpublished Doctoral dissertation, University of Edinburg.

[47] Hollos, D., Karl, H., & Wolisz, A. (2004). Regionalizing Global Optimization Algorithms to Improve the Operation of Large Ad Hoc Networks. Proceedings of the IEEE Wireless Communications and Networking Conference, Atlanta, Georgia, USA.

[48] Katzela, I., & Naghshineh, M. (1996). Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. IEEE Personal Communications, 10–31.

[49] Salgado, H., Sirbu, M., & Peha, J. (1995). Spectrum Sharing Through Dynamic Channel Assignment For Open Access To Personal Communications Services. Proc. of IEEE Intl. Communications Conference (ICC), pp. 417-22.

[50] Tudor Palade and Emanuel Puschita, "Requirements for a New Resource Reservation Model in Hybrid Access Wireless Network", WSEAS Tans. On Communications, Issue 3, Volume 7, March 2008, pp 144-151.