On the Efficient Implementation of a High Performance Multi-Agent Simulation System for Modeling Cellular Communications Involving a Novel Event Scheduling Algorithm

P.M.Papazoglou
University of Portsmouth/ECE Department, Portsmouth, UK and Lamia Institute of Technology, Greece

Dimitrios A. Karras
Chalkis Institute of Technology, Automation Dept. Chalkis, Greece, emails: dakarras@ieee.org, dakarras@teihal.gr

R.C.Papademetriou
University of Portsmouth/ECE Department, Portsmouth, UK

***Abstract*—Simulation models are used in the design, development and evaluation of wireless communication systems. While basic network entities as Base Stations and users are always taken into account in such simulation modeling, a critical but currently underestimated factor that affects the simulated network behavior is the corresponding simulation model which represents the physical activities and the events of the network. A novel event scheduling mechanism for supporting concurrent network events and a novel network modeling methodology based on the multi-agent concept implemented through multi-threading technology is presented in this paper. The state of the art event scheduling mechanism supports only sequential events and on the other hand, the multi-agent technology has been used only for modeling network nodes. Additionally, the technical aspects of the multi-threading technology regarding the agent implementation and scheduling for simulating wireless communication systems has not been investigated so far in the literature. The proposed simulation framework in this paper gives improved solutions to the above issues.**

Keywords- Event scheduling; multi-agents; network modeling; wireless network simulation

I. INTRODUCTION

An agent can be defined as an autonomous computational system that works for specific and predefined goals [1-3]. Moreover, an agent interacts with the surrounding environment and acts on it. The most known and important attributes of an agent are:

- Adaptability (agent change according to external or internal events) [4,5]
- Autonomy (control of its own actions) [6-8]
- Collaboration (with other agents for achieving common goals)
- Interactivity (with surrounding environment)

According to [9], a Multi-Agent System (MAS) consists of a number of agents which interact through communication.

These agents act in an environment within which they have different areas of influence (fig. 1). Within the environment many influence areas may coincide. MASs can be viewed as a loosely coupled network of problem-solver entities [10] that collaborate together with the common goal to solve the whole complex problem beyond the solving capabilities of each individual entity. Agent technology has been also used in the management of telecommunication systems [11-14].

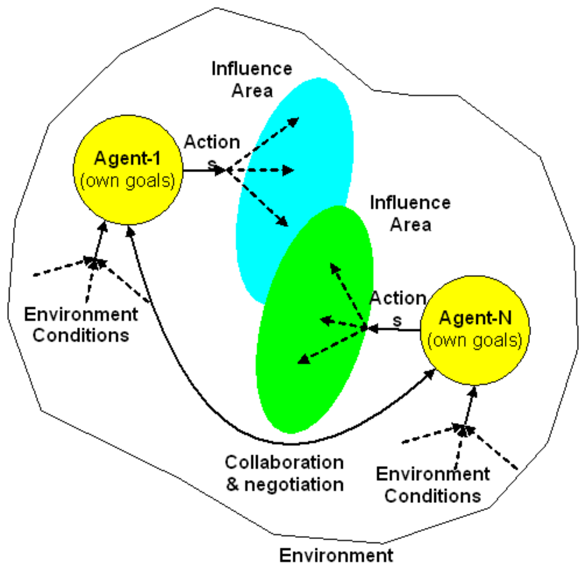

Fig. 1 A multi-agent environment

A. *Multi-Agent Systems for solving the resource allocation problem in wireless communications*

The resource allocation problem constitutes one of the most critical issues in wireless communication systems [15-18]. An overview of agent technology in communication systems is presented by [11]. The majority of existing studies are focused on node oriented approaches [11-14]. According to these approaches, the modeling methodology is based on the network nodes such as base stations and cells. For example, in [13] the final decision for call admission is based on the participation of neighbour cells which are based on agents that represent cells or BSs (Base Stations).

A cooperative negotiation in a MAS for supporting real-time load balancing of a mobile cellular network is described in [14]. A distributed channel allocation scheme based on intelligent software agents is proposed by [12] where a comprehensive simulation model for wireless cellular networks has been built. In [12], intelligent collaborative software agents give autonomy to BSs, increase the network robustness, allow negotiation of network resources and improve resource allocation.

A different approach on using the multi-agent methodology for resource allocation in wireless systems, based on handling concurrency through real time event scheduling mechanisms is introduced in [18], and the present paper follows the same line of research extending in depth the methodology and the results presented there initially.

B. Network Event scheduling in modern simulation environments

Discrete Event Simulation (DES) [19-28] is the most known technology for building simulation models especially for wireless communications. In this technology, events are happening at discrete points in time. The real network activities are modeled by event generators and a specialized mechanism called scheduler sends these events for later execution based on the corresponding time stamps (generation time). The above time stamps define accurately the execution sequence of the enqueued events (unique time stamps). The Calendar Queue (CQ) concept which has been firstly introduced by [29] represents the state of the art event scheduling mechanism and is used by the most known simulation tools such as ns-2 (Berkeley, USA) [30] which is adapted by the 44.4% of the scientific community [31].

This approach is based on the ordinary desk calendar where one page is dedicated for each day. Every inserted event it is scheduled for execution at a later time. The highest priority event (lower time stamp) on the calendar is fetched by searching the page for today's date and removing the earliest event (lower time stamp) written on that page [29]. A CQ is implemented with an array of lists inside the computer. The N events are partitioned within M shorter lists called Buckets. Each bucket is associated with a specific range of time stamps (priority range) corresponding to future events. An 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

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

For locating the bucket number m(e) where an event e will occur at time t(e) the following type is used :

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

As an example, if M=8, N=10, δ=1 and t(e)=4.68, then m(e)=4.

Figure 1 shows a complete operation of the CQ in a sequential DES system which simulates wireless network services such as new call, reallocation (handoff), etc. Initially, the CQ contains 6 events (step 0) to be executed at a later time. The event e1 (NC - New Call) with time stamp 3.62 will be dequeued first (step 1) because it has the highest priority (minimum time stamp) among the buckets of the queue and will be executed first (step 2, figure 2). In step (3), a new event with time stamp 9.98 is enqueued and the event e2 (RC - Reallocation Call) is dequeued in step (4). The whole operation of the CQ follows the rule Dequeue-Execute-Enqueue. This rule can be evaluated in any step of the whole operation (e.g. Dequeue {step(9)}-Execute{step(10)}-Enqueue {step (11)}, fig. 2 and 3). The event generator produces events with corresponding time stamps. The dequeued events are executed in a descending priority order or in ascending time stamp order (fig. 3). Even for events with almost equal time stamps the final execution is sequential over simulation time.

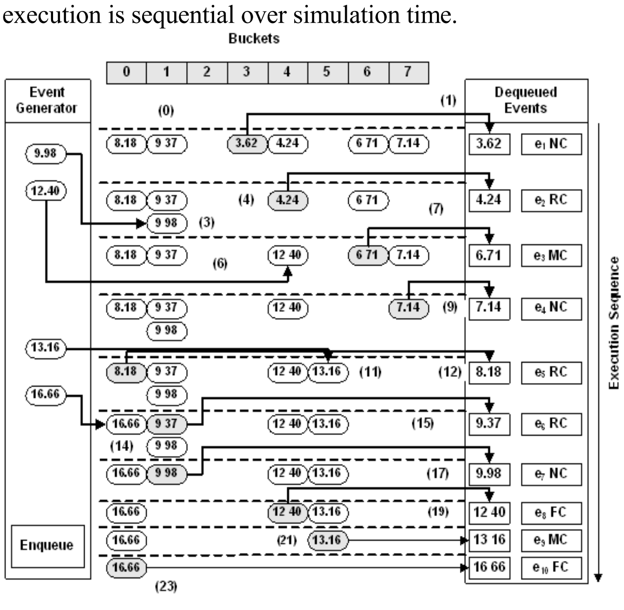

Fig. 2 Sample of a Sequential DES operation (Events/Services: NC-New Call, RC-Call Reallocation/Handoff, MC-Movement Call, FC-Finished Call)

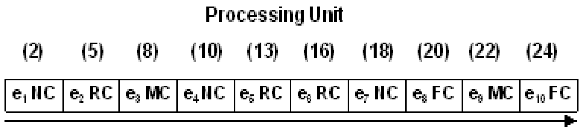

Fig. 3 Event execution based on dequeue order (Events/Services: NC-New Call, RC-Call Reallocation/Handoff, MC-Movement Call, FC-Finished Call)

II. THE PROPOSED SIMULATION MODEL

A. Modeling methodologies

A.1 Services oriented modeling based on the multi-agent technology

Due to network services concurrency and autonomy inside a cellular network, there are several common attributes between services and agents. Table 1 shows the most important attributes.

Basic Network service behaviour	Basic Agent behaviour
Adapt network service behaviour to MU calls (e.g. give call priority)	Adaptability
Make decision for its own goal (e.g. RCA for minimizing the number of dropped calls)	Autonomy
Communication with other service procedures for balancing network performance	Collaboration
Gathers information from wireless network environment and acts on it	Interactivity

Table 1. Common attributes between network services and agents

Based on the above attributes, the supported network services have been modeled as agents. On the other hand, there is lack of suitable representations for such network procedures in the simulation systems so far in the literature.

A.2 Supported network services by the proposed simulation model

Four basic network services are supported by the experimental simulation model:

- New call arrival (NC)
- Reallocation check (RC)
- User Movement (MC)
- Call termination (FC)

New calls are generated by the Poisson distribution and can be viewed as aperiodic events. The aperiodic task arrival times can be modeled as Poisson process. Additionally, Each day of network operation is divided into five zones according to traffic conditions.

A.3 Multi-Agent Layered architectural model

The whole simulation model is organized in a layered architecture based on the corresponding functionality. The above mentioned network services are now represented by agents inside the layered simulation model. Figure 4 shows the layered model.

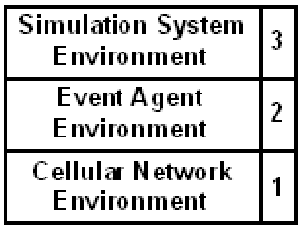

Fig.4 Layered simulation model

The corresponding layer functionality is as follows:

- *Layer 1* (core). It represents the cellular network structure (base stations, cells, signal environment, etc), where the events take place.
- *Layer 2.* It contains the four agents (for each cluster) and describes the network behaviour for the service support.
- *Layer 3.* It is a Control Agent that synchronizes the actions of the four agents (agent activation and deactivation at the beginning and at the end of each simulation time respectively).

A.4 The novel event scheduling mechanism adapted to current network conditions

The proposed mechanism is called Priority Queue-Time Division Multiplexing (PQ-TDM), constitutes an extension of the state of the art (CQ) approach and is initially introduced in [32]. The core architecture of the mechanism is similar to CQ and consists of an array of lists where each list contains future events with individual priorities. The list of N concurrent/non-concurrent events are partitioned to shorter lists called Priority Buckets. The simulation model implementation is based on two thread categories which are: (a) network events (single threads) and (b) a special thread (Time Clock Thread - TCT). The maximum priority (PMAX) is assigned to TCT as compared to the rest of the threads. If the Multi Threaded Platform supports P priorities, then DP is the priority distance between priority associated with the priority bucket having the maximum priority and PMAX priority of the TCT. If the supported priorities are {1,2,…P} and PMAX=P, then PMAX - DP are the supported buckets in which the event list is partitioned. Finally, each bucket is associated with a specific range of priorities. An 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

$$p(e)\in[(p(P_{MAX}-D_p)+m)\delta,(p(P_{MAX}-D_p)+m+1)\delta] \quad (3)$$

For calculating the bucket number m(e) where an event e will be stored with priority p(e) the following formula is introduced:

$$m(e) = \left\lfloor \frac{p(e)}{\delta} \right\rfloor \bmod (P_{MAX} - D_P) \qquad (4)$$

Regarding time t(e) of the event-thread e, it should be remarked that now it is determined by the Multi Threaded Simulation Platform by a Time Division Multiplexing (TDM) procedure. The TDM procedure is based on the Time slice ΔT which represents the basic entity. 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.

A.5 PQ-TDM mechanism operation

Let NCp, RCp, MCp, and FCp the priorities of the corresponding events NC, RC, MC and FC respectively. If NCp=3, RCp=1, MCp=1, FCp=1, PMAX=10, and N=3, then the resulted buckets are as follows (table 2):

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

Table 2. Bucket structure (Pri=Priority, B.No=Bucket number, Multi=Time slice multiplier)

According to table 1, the execution starts with the first NC event (NC1) for time ΔT x 3 (TSW=Time Slice Width) and after that time the event FC1 is executed next for ΔT x 1 time, and so on. Table 3 shows the execution interleaving according to the example of table 1.

	Execution Sequence					
Time	x 3	x 1	x 1	x 1	x 3	x 1
Event	NC_1	FC_1	RC_1	MC_1	NC_2	RC_1

Table 3. Sample of event execution

A.6 The PQ-TDM mechanism in a large scale environment

When the network under investigation is a large scale network, the proposed scheduling mechanism has to be distributed in the whole network. Assuming that the cellular network has N cells distributed in cell clusters where each cluster contains i cells, the total number of clusters is N/i. Each set of the four threads and the corresponding event scheduler is duplicated in every cluster. Thus, the total required threads are 4*(N/i). On the other hand, the simulation model consists of (N/i) PQ-TDM schedulers (subsets) and one main PQ-TDM scheduler (superset) (fig. 5). When two or more processors are available, multiple hierarchical levels of event scheduling can be used.

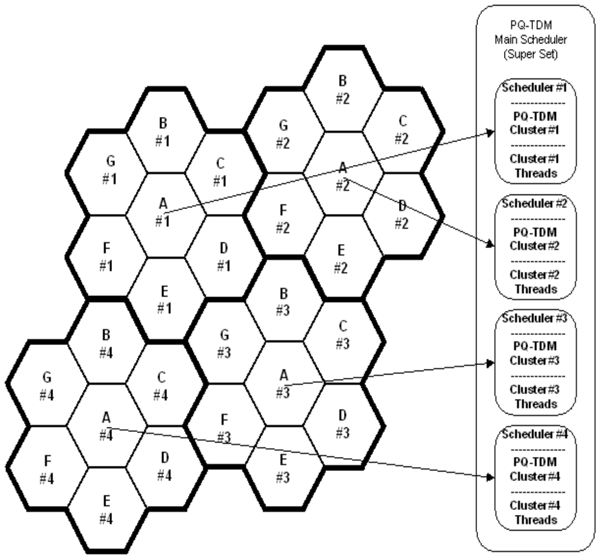

Fig. 5 Distributed schedulers, one main scheduler

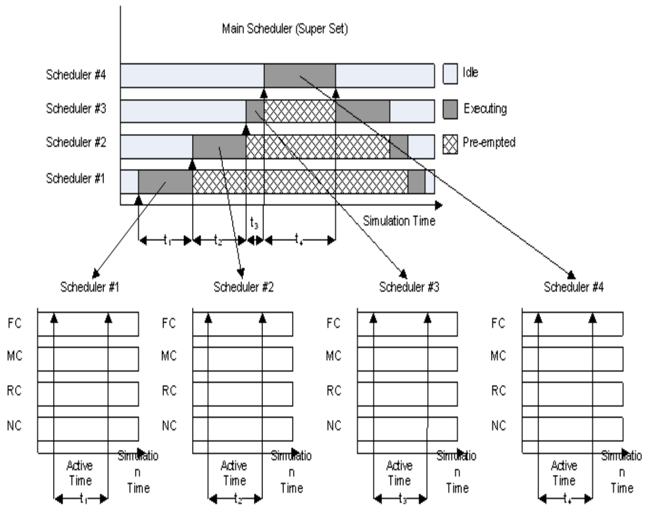

Fig. 6 Activation time of the distributed schedulers

According to figure 6, the simulation time is given to main scheduler and this time is distributed among the single schedulers (subsets). This mechanism (fig. 6) can be called multi-scheduling due to the fact that the simulation time is distributed among schedulers like CPU time which is distributed among active threads.

A.7 Multi-agent negotiation strategy

The RC Agent (RCA) is responsible for the success of hand-off attempts and the NC Agent (NCA) is responsible for the new call admission. The network performance is measured with dropping and blocking probability which represent the successfulness of the network for supporting handoffs and call admissions respectively. When the current network performance is bellow the desired blocking or dropping levels, the corresponding network agents must take the initiative to improve the network performance.

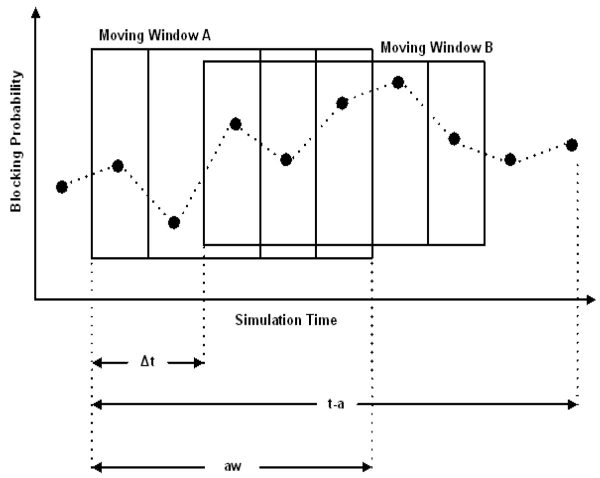

Fig. 7 measuring the network behavior progress

Due to the limited channel availability, the network agents have to negotiate in order to find a suitable solution to that problem. If both metrics (blocking, dropping) are at critical level, the negotiation is competitive otherwise is cooperative. The above negotiation is based on predefined rules and is implemented through agent dialog and message exchange. The network performance behaviour is measured by calculating the blocking or dropping probability progress among current and previous simulation steps. More precisely, two moving windows are used in order to collect information about the behaviour progress. Figure 7 shows the two moving windows among the network performance points.

NCA Status	Description
1 (Good)	Std Blocking Pr.(W_B) < Std Blocking Pr.(W_A)
0 (Stable)	Std Blocking Pr.(W_B) = Std Blocking Pr.(W_A)
-1 (Critical)	Std Blocking Pr.(W_B) > Std Blocking Pr.(W_A)

Table 4. NCA Status description

Window A starts at t-a and ends at t-a+aw and window B starts at (t-a)+Δt and ends at (t-a+aw)+Δt. The corresponding ratios are calculated between the two moving windows. Tables 4 and 5 show the corresponding agent status based on the moving windows calculations.

Each agent has a mail box for receiving requests from other agents. For example, if the RCA is in critical condition, then sends a message to NCA for priority decrement in order to improve the handoff performance. In the same way, the NCA communicates via messages with RCA for improving the corresponding performance inside its influence area. Figure 8 shows how the two basic network agents can exchange messages.

RCA Status	Description
1 (Good)	Std Dropping Pr.(W_B) < Std Dropping Pr.(W_A)
0 (Stable)	Std Dropping Pr.(W_B) = Std Dropping Pr.(W_A)
-1 (Critical)	Std Dropping Pr.(W_B) > Std Dropping Pr.(W_A)

Table 5. RCA Status description

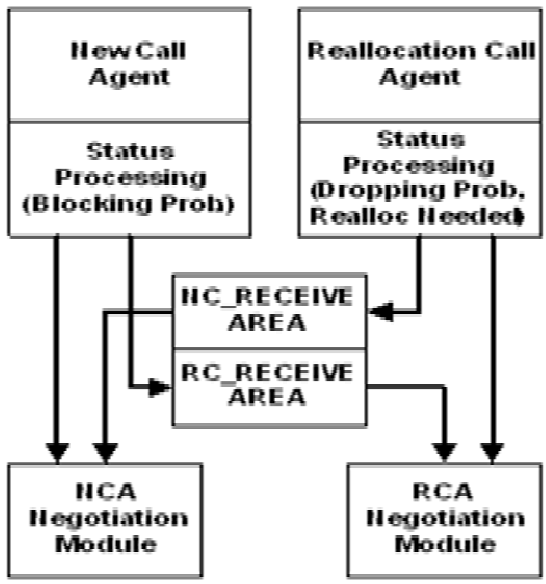

Fig. 8 Agent communication

An agent takes decisions not only for its self but also for response to any incoming messages. If the self status is not critical and the current priority is not minimum, then a possible request for priority decrement from another agent is accepted. Figure 9 shows the algorithm for implementing the decision making procedure of each agent.

B. Implementation issues

B.1 Algorithmic implementation of the proposed PQ-TDM mechanism

Figure 10 shows the proposed event scheduling algorithm for one network cluster.

```
BEGIN {0}
  If (inbox_message) {1}
    If (self_status==-1) {2}
      If (self_priority<max_priority) {3}
          Increase_self_priority
      End_if {3}
      Message("request rejected")
    Else {2}
      If (self_priority>min_priority) {4}
          Decrease_self_priority
          Message("request accepted")
      Else {4}
          Message("request rejected")
      End_if {4}
    End_if {2}
  Else {1}
    If (self_status==-1) {5}
      If (self_priority<max_priority) {6}
          Increase_self_priority
      End_if {6}
    End_if {5}
  End_if {1}
END {0}
```

Fig. 9 Agent decision making algorithm

```
//Cluster #n
//AST=Assigned Time by the main scheduler
//The following data are applied only for current cluster
MAEP=Maximum Allowed Event Priority
PMAX=Maximum Priority
Priority set={1,2,…,PMAX}
DP=PMAX-MAEP //priority distance
N=PMAX-DP=MAEP //number of buckets
NCp, RCp, MCp, FCp = event priorities
δ=1 //bucket width
Create N Buckets // Bucket[N]
If new event p(e) exists
   m(e)=Round(p(e)/δ) mod MAEP
   Store event [p(e)] → Bucket_tail[m(e)] //Enqueue
End if
While PQ != empty
  Loop i=(N-1) Downto 0
    Fetch event from Bucket_top[i];
    Execute event for time AST x i
  End Loop
End while
```

Fig. 10 The PQ-TDM algorithm for one cluster

According to the above algorithm (fig. 10), a generated event is stored in the corresponding bucket based on its priority and the bucket structure. Additionally, the algorithm searches for the next event (priority based search) and sends this event for execution. The execution duration (active event period) is defined by the main scheduler.

B.2 Building a controlled scheduler under the JVM unpredictability

An internal mechanism of JVM, called scheduler, defines the real-time order of thread execution. Scheduling can be controlled by the programmer and is categorized as follows:

- Non pre-emptive
- Pre-emptive

In non pre-emptive, the scheduler runs the current thread forever and requires from this thread to tell explicitly if it is safe to start another thread. In pre-emptive, the scheduler runs a thread for a specific time-slice (usually a tiny period within a second) and then "pre-empts" it, (calling suspend()), and resumes another thread for the next time-slice. The non pre-emptive scheduling can be very useful especially in time critical (e.g. real time) applications when the interruption of thread execution can happen in the wrong moment. Modern schedulers are usually pre-emptive, therefore the development of MT applications is easier. JVM uses priorities for scheduling threads. Initially, it gives equal priorities to all threads. A major drawback of the JVM is that the behavior of the scheduler (e.g. thread execution sequence) can not be predicted [33]. For that reason, only a controlled scheduling mechanism is proposed and applied in this study.

As mentioned before, the thread execution sequence can not be predicted under the JVM specification. On the other hand, the thread execution sequence is critical regarding the supported network services as these threads constitute the implementation of the agents that represent the offered services by the network. In order to overcome the JVM unpredictability, a second level scheduling mechanism has been developed under the default JVM mechanism. The thread control based on the JVM instructions is time consuming and can give also undesired results. The proposed methodology controls the thread code activation and not the thread it self. Thus, simple conditions are used that permit or not the thread core code execution. A thread remains active but the internal core code (program instructions) is activated only under specific conditions.

Figure 11 shows the pseudo code of clock implementation inside the main scheduler.

```
Thread main scheduler CLOCK
//CS#n=Cluster Scheduler #n
//CS#n=Cluster n Scheduler priority
{
  Main action
   {
     While (simulation time step)
          {
           CS#1_active=1;
           Sleep(CS#1_limit);
           CS#1_active=0;

           CS#2_active=1;
           Sleep(CS#2_limit);
           CS#2_active=0;

           CS#3_active=1;
           Sleep(CS#3_limit);
           CS#3_active=0;

           CS#4_active=1;
           Sleep(CS#4_limit);
           CS#4_active=0;
          }
   }
}
```

Fig. 11 Clock inside the main scheduler (super set)

Figure 12 shows the cluster scheduler implementation. The cluster scheduler n remains active for CS#n time period which is defined by the main scheduler clock.

```
Cluster#n_Scheduler
{
  Main action
    {
      While (AST)
          {
           If (CS#n_active==1)
             {
               //Thread CLOCK core code
             }
          }
    }
}
```

Fig. 12 Cluster scheduler activation

Figure 13 shows how the network agents (threads) are activated by the clock inside a cluster.

Using the Thread.start() method, all the threads become active but the core code of the threads (local schedulers and network services) except main scheduler clock is disabled while CS#n_active=0. The thread main scheduler CLOCK is under execution in most of the times because its priority has the maximum value as compared to the rest of the threads. After CS#1_limit time the corresponding core code is deactivated.

```
Thread CLOCK core code
{
  Main action
   {
     While (simulation time step)
          {
           Thread#1_active=1;
           Sleep(Thread#1_limit);
           Thread#2_active=0;
           …
           …
           Thread#n_active=1;
           Sleep(Thread#2_limit);
           Thread#n_active=0;
          }
   }
}
```

Fig. 13 Thread control inside cluster

Table 6 shows the two level scheduling mechanisms.

Despite the JVM scheduler unpredictability, the thread execution sequence is controlled by a second level scheduler which is based on a main clock. For guarantee the execution sequence T1 (3ΔT), T3 (2ΔT), T2 (2ΔT), the time clock defines the thread code activation sequence. If the JVM default scheduler gives an execution time slice to another thread (undesired thread), the corresponding code is blocked by the time clock. Thus, the thread execution (at code level) sequence is controlled through the second level scheduler. The only drawback of the proposed method is that when the JVM assigns an execution time slice to another thread, this time is wasted. This drawback is not significant due to the fact that the time slices represent too small time periods.

Scheduler	Threads									
JVM	T2	T1	T1	T2	T1	T3	T2	T1	T3	T2
Time Clock	X	T1	T1	X	T1	T3	X	X	T3	T2

Table 6 Execution sequence : T1 (3ΔT), T3 (2ΔT), T2 (1ΔT), X=Inactive Thread code

III. SIMULATION MODEL VALIDATION

The implemented algorithms of the proposed mechanisms (e.g. PQ-TDM) in this study have been tested in an elementary simulation environment that integrates the basic network and simulation components. The validation procedure of the simulation environment consists of three validation levels which are:

- Calendar Queue scheduling mechanism implementation
- Network environment (signal propagation, interference and signal measurements)
- Simulated network performance compared to theoretical computations

A. CQ Algorithm validation

The CQ scheduler implementation within ns-2 can be found in [29]. The implementations of the CQ in this study are based on these algorithms. The CQ algorithms define two individual operations which are:

- CQ operation (Enqueue, Dequeue)
- CQ internal functionality (creation, initialisation, resize, etc)

Figure 14 shows the corresponding pseudo code for the enqueue operation [29].

```
1 /* This adds one entry to the queue. */
2 {
3 int i;
4 /* Calculate the number of the bucket in which to
place the new entry. */
5 i = priority/width; / * Find virtual bucket. */
6 i = i % nbuckets; /* Find actual bucket. */
7 Insert entry into bucket i in sorted list;
8 ++qsize; / * Update record of queue size. */
9 /* Double the calendar size if needed. */
10 if (qsize > top-threshold) resize(2 * nbuckets);
11 }
```

Fig. 14 C-pseudo code, Enqueue operation [29]

In line 5 of figure 14, the fraction t(e)/δ is calculated (see equation 2). Finally, the number of bucket to store the new generated event is calculated (t(e)/δ) mod M (see equation 2). It is obvious from figure 14 that the variable priority represents the time stamp of the generated event.

The corresponding Java-pseudo code for the enqueue operation is as shown in figure 15.

The corresponding implementations of the [29] pseudo code within ns-2 (in C) can be found in [34].

The dequeue operation has been implemented similarly based on the algorithm of [29].

```
/* This adds one entry to the queue. */
{
int i;
/* Calculate the number of the bucket in which to
place the new entry. */
i = Math.floor(priority/width); / * Find virtual
bucket. */
i = i % nbuckets; /* Find actual bucket. */
Insert entry into bucket i in sorted list;
++qsize; / * Update record of queue size. */
/* Double the calendar size if needed. */
if (qsize > top_threshold) resize(2 * nbuckets);
}
```

Fig. 15 Java-pseudo code, Enqueue operation in the evaluated model

B. Network environment validation

The implemented environment has been built on the known theoretical components for radio propagation, signal measurements and cellular network operation. The validation of this environment is necessary in order to prove the correctness of the results. The validation procedure can be found also in [35] and consists of two phases which are:

- Monte Carlo simulations
- pdf evaluation based on theoretical solutions

C. Network performance validation

Blocking and dropping probability constitutes the most popular and applicable performance metrics for network behavior, especially for channel allocation and bandwidth management [36-43].

According to [38], blocking and dropping probabilities can be defined as follows:

- Call Blocking probability, P_B = probability that a new call is denied access to the network.
- Call Dropping or Forced Termination probability, P_D = probability that a call in progress is forced to terminate earlier.

The blocking probability $P_{blocking}$ is calculated from the ratio

$$P_{blocking} = \frac{number\ of\ blocked\ calls}{number\ of\ calls} \tag{5}$$

The dropping probability $\mathbf{P_{fc}}$ is calculated from the ratio

$$P_{fc} = \frac{number\ of\ \ forced\ calls}{number\ of\ calls - number\ of\ blocked\ calls} \tag{6}$$

The blocking probability can be also theoretically calculated. If the received power of each MU is high enough, it is assumed that the interference from other MUs can be ignored. The theoretical formula is as follows:

$$P_{blocking\_theoretical} = \frac{\binom{n-1}{s}(vh)^s}{\sum_{i=0}^{s}\binom{n-1}{i}(vh)^i} \quad (7)$$

where n is the number of users, s is the number of channels, v is the average call arrival rate (for no connected MU) and h is the average call holding time.

Equation 7 does not take in to consideration critical factors that affect the blocking probability such as traffic conditions, service type, channel allocation strategy, etc. Figure 16 shows the theoretical blocking probability that derives from eq. 7 as compared to simulated blocking probability.

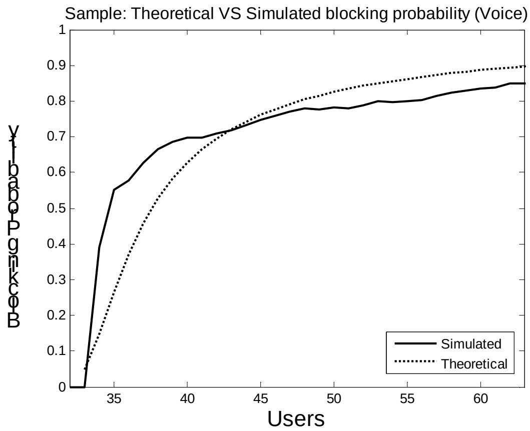

Fig. 16 Theoretical blocking probability versus simulated

IV. SIMULATION RESULTS

The simulation results are categorized as follows:

- Thread execution control (showing the effectiveness of the proposed event scheduling mechanism)
- Negotiation dialog (event priority adaptation to current network conditions)
- Multi-agent modelling (proposed modelling based on multi-agent technology versus single multi-threaded approach)

A. *Thread execution control*

The simulation results show that the proposed mechanisms control effectively the activation of the distributed schedulers based on the time distribution by the main scheduler. In the same results, the thread execution unpredictability is also shown. Figures 17 and 18 show the thread execution sequence unpredictability caused by the default JVM scheduler.

MMMMMMMMMMMMMMMMMMMMMMMMMMMN
FRNFRMNFRMNFRMNFRMNFRMNFRMNFRMNFRMNFR
MNFRMNFRMNFRMNFRMNFRMNFMNFMNRMNR
FMRFMRFMRFMRFMRFMRFMRFMRFMFMRFMRF
MRFMRFMRFMRFMRFMNFMNRFNRFNRFNRFNR
FNRFNRFNRFNRFNRFNRFNRFNRFNRFNRFN
RFMRFMNRMNRMNRMNRMNRMNRMNRMNRM
NRMNRMNRMNRMNRMNRMNRMNRMNRFNRFM
NFMNFMNFMNFMNFMNFMNFMNFMNFMNF
MNFMNFMNFMNFMNFMNRMNRFMRF …

Fig. 17 Thread execution sequence, sample #1, Default JVM scheduling

MRMRMRMRMRMRMRMRMRMRMRMRMFNRNRNR
NRNRNRNRNRNRNRNRNRNRNRNRNRFMRMRM
FNMNMNMNMNMNMNMNMNMNMNMNMNFRN
RNRNRNRNRFMRMRMRMRMRMRMRMRMRMR
MRMRMFNMNMNMNMNMNFRNRNRNRNRNR
NRNRNRNRNRNRFMRMRMRMRMRMFNMNMNM
NMNMNMNMNMNMNMNMNMNFRNRNRNRNRNR
FMRMRMRMRMRMRMRMRMRMRMRMRMFNMN
MNMNMNMNFRNRNRNRNRNRNRNRNRNRNRNR
NRFMRMRMRMRMRMFNMRNMRNMRNMRNMR
NMRNMR …

Fig. 18 Thread execution sequence, sample #2, Default JVM scheduling

Figure 19 shows an example of the distributed scheduler activation by the main scheduler. Additionally, figure 20 shows the network event execution sequence inside a cluster.

#1#1#1#2#3#4#4#1#1#1#2#3#4#4#1#1#1#2#3#4#4
#1#1#1#2#3#4#4#1#1#1#2#3#4#4#1#1#1#2#3#4#4
#1#1#1#2#3#4#4#1#1#1#2#3#4#4#1#1#1#2#3#4#4
#1#1#1#2#3#4#4#1#1#1#2#3#4#4#1#1#1#2#3#4#4
…

Fig. 19 Local scheduler activation (4 cluster sample with CS#1=3, CS#2=1, CS#3=1, CS#4=2 and activation sequence CS#1#2#3#4)

NNNRFMMNNNRFMMNNNRFMMNNNRFMM
NNNRFMMNNNRFMMNNNRFMMNNNRFMM
NNNRFMMNNNRFMMNNNRFMMNNNRFMM
NNNRFMMNNNRFMMNNNRFMMNNNRFMM
...

Fig. 20 Network event execution inside cluster (with NCp=3ΔT, RCp=1ΔT, FCp=1ΔT, MCp=2ΔT)

B. *Negotiation dialog*

Table 7 shows how the agent priorities are changing according to the current network needs and the negotiation dialog results. Assume that the initial agent priorities are 6 and 2 for RCA and NCA respectively. These priorities correspond to the time assignments ΔT for the thread execution. At simulation step 70, the RCA sends a message (request) to NCA for priority decrement. This request is rejected due to the fact that the NCA status is critical. After the rejection, the RCA takes the decision for self priority increment and thus the priority level changes from 6 to 7. At step 100, the RCA request is accepted from NCA and the NCA decreases its priority to help RCA to improve its influence to the wireless network environment. Finally at step 120, both requests from the RCA and NCA agents are rejected and the agents take decision for its status.

Sim step	Message (priority decrement)		Res-ponse	RCA Priority	NCA Priority
	From	To			
…				(6) 6 x ΔT	(2) 2 x ΔT
70	RCA	NCA	Rejected NCA Status -1	(7) 7 x ΔT	(2) 2 x ΔT
80	-	-	-	(7) 7 x ΔT	(2) 2 x ΔT
90	-	-	-	(7) 7 x ΔT	(2) 2 x ΔT
100	RCA	NCA	Accepted	(8) 8 x ΔT	(1) 1 x ΔT
110	RCA	NCA	Rejected NCA priority minimum	(9) 9 x ΔT	(1) 1 x ΔT
120	NCA	RCA	Rejected RCA status -1	(10) 10x ΔT	(2) 2 x ΔT
	RCA	NCA	Rejected NCA Status -1		

Table 7. Negotiation dialog

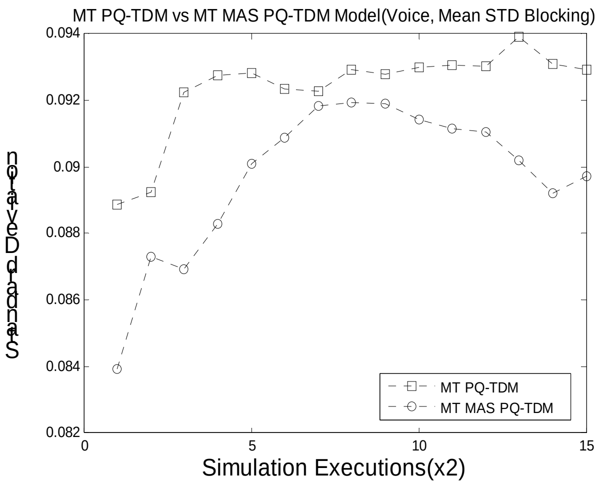

Fig. 21 Multi Threaded PQ-TDM Versus Multi Threaded Multi Agent PQ-TDM model (Voice, Mean STD Blocking Probability, Classical DCA)

C. Multi-agent modeling

Figures 21 and 22 show the simulation model behavior in terms of standard deviation for blocking and dropping probabilities respectively.

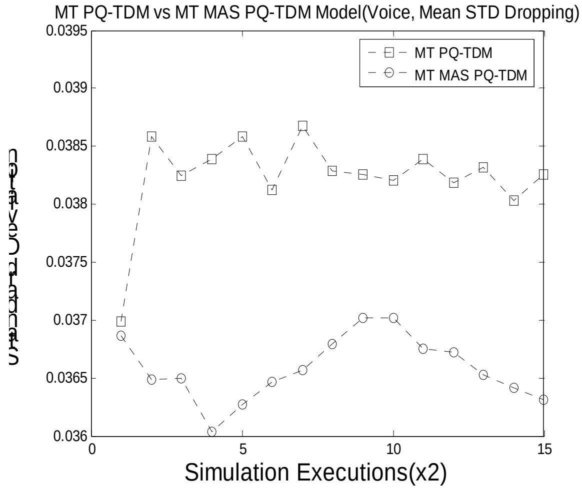

Fig. 22 Multi Threaded PQ-TDM Versus Multi Threaded Multi Agent PQ-TDM model (Voice, Mean STD Dropping Probability, Classical DCA)

V. CONCLUSIONS

A novel simulation model for wireless communication systems has been presented in this paper. This model supports both sequential and concurrent network events and constitutes an extension of the Calendar Queue approach. The proposed simulation model has been developed after a thorough investigation of the involved technologies and methods such as multi-agents, event scheduling, time division multiplexing and multi-threading. Multi-agents are used for modeling network services in a services oriented approach. On the other hand, the proposed event scheduling mechanism is mainly based on the multi tasking theory and the agent implementation using multi-threading technology. It is obvious that for developing a scheduler for concurrent network events, suitable model architecture must be used. Moreover, the multi-threaded platform such as JVM introduces several important drawbacks such as thread execution unpredictability and deadlocks in the case of shared data access from multiple threads. The proposed algorithmic implementation of the PQ-TDM mechanism overcomes the default JVM unpredictability but involves wasted time periods if the active thread controlled by the default JVM scheduler does not match with the desired active thread by the proposed scheduler. From the above findings it is clear that a new JVM specification regarding the thread execution sequence must be developed.

REFERENCES

[1] Maes, P. (1995). Artificial Life Meets Entertainment: Life like Autonomous Agents. Communications of the ACM, 38 (11), 108-114.

[2] Smith, D.C. (1994). KidSim: Programming Agents Without a Programming Language. Communications of the ACM, 37 (7), 55-67.

[3] Hayes-Roth, B. (1995). An Architecture for Adaptive Intelligent Systems. Arti-ficial Intelligence: Special Issue on Agents and Interactivity, 72, 329-365.

[4] Splunter, S., Wijngaards, N., & Brazier, F. (2003). Structuring Agents for Adaptation. In E. Alonso et al. (Eds.), Adaptive Agents and Multi-Agent Systems (pp. 174-186), LNAI, Vol. 2636.

[5] Russell, S., & Norvig, P. (2002). Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall.

[6] Huhns, M., & Singh, M. (Eds.). (1998). Agents and Multiagent Systems: Themes, Approaches, and Challenges. Readings in Agents, (pp. 1-23). USA: Morgan Kaufmann Publishers.

[7] Norman, T., & Long, D. (1995). Goal Creation in Motivated Agents. Wooldridge, Jennings (Eds.), Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890).

[8] Ekdahl, B. (2001). How Autonomous is an Autonomous Agent? Proceedings of the 5th Conference on Systemic, Cybernetics and Informatics (SCI 2001). Orlando, USA.

[9] Jennings, N.R. (2000). On agent-base software engineering. Artificial Intelligence, 117, 277-296.

[10] Sycara, K. (1998). Multi-Agent Systems. Artificial Intelligence Magazine, 19(2).

[11] Hayzelden, A., & Bigham, J. (1999). Software Agents for Future Communications Systems. Berlin: Springer-Verlag.

[12] Bodanese, E.L. (2000). A Distributed Channel Allocation Scheme for Cellular Networks using Intelligent Software Agents. Unpublished Doctoral dissertation, University of London, UK.

[13] Iraqi, Y., & Boutaba, R. (2000). A Multi-agent System for Resource Management in Wireless Mobile Multimedia Networks, LNCS 1960, pp. 218–229, Springer-Verlag Berlin Heidelberg.

[14] 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.

[15] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "On the Implementation of Ant Colony Optimization Scheme for Improved Channel Allocation in Wireless Communications", IEEE International Conference on Intelligent Systems, 2008

[16] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou "An Improved Multi-Agent Simulation Methodology for Modelling and Evaluating Wireless Communication Systems Resource Allocation Algorithms", Journal of Universal Computer Science, 2008, vol.14,issue 7,pp. 1061-1079

[17] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "A Critical Overview on the Recent Advances in Channel Allocation Strategies for Voice and Multimedia Services in Wireless Communication Systems and the Applicability of Computational Intelligence Techniques", Proceedings of the 10th WSEAS International Conference on Mathematical Methods, Computational Techniques And Intelligent Systems (MAMECTIS '08)

[18] P.M.Papazoglou, D.A.Karras, R.C.Papademetriou, "A Multi-Agent Simulation Model for Wireless Communications Involving an Improved Agent Negotiation Scheme Based on Real Time Event Scheduling Mechanisms", Computer Modeling and Simulation, 2008. EMS '08. Second UKSIM European Symposium on 8-10 Sept. 2008 Page(s):246 - 252

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

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

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

[22] 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).

[23] 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.

[24] 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.

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

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

[27] 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.

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

[29] 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.

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

[31] Kurkowski, S., Camp, T., & Colagrosso, M. (2005). MANET Simulation Studies: The Current State and New Simulation Tools.

[32] 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

[33] Mengistu, D., Lundberg, L. and Davidsson, P. (2007). Performance Prediction of Multi-Agent Based Simulation Applications on the Grid. international journal of intelligent technology volume 2 number 3 2007 ISSN 1305-6417

[34] http://www-rp.lip6.fr/ns-doc/ns226-doc/html/index.htm

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

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

[37] Wong, S.H. (2003). Channel Allocation for Broadband Fixed Wireless Access Networks. Unpublished doctorate dissertation, University of Cambridge.

[38] 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.

[39] 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).

[40] Tripathi, N.D., Jeffrey, N., Reed, H., & VanLandingham, .F. (1998). Handoff in Cellular Systems. IEEE Personal Communications.

[41] 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.

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

[43] Lee, W.C., Lee, J., & Huff, K. (1999). On Simulation Modeling of Information Dissemination Systems in Mobile Environments. LNCS 1748, 45–57.