A Component Modular Modeling Approach based on Object Oriented Petri Nets for the Performance Analysis of Distributed Discrete Event Systems

Aladdin Masri Thomas Bourdeaud'huy Armand Toguyeni
LAGIS – CNRS UMR 8146
Ecole Centrale de Lille
59651 Villeneuve d'ASCQ, France
{aladdin.masri, thomas.bourdeaud_huy, armand.toguyeni@ec-lille.fr}

Abstract

Distributed Discrete Event Systems (Distributed DES) are increasing with the development of networks. A major problem of these systems is the evaluation of their performance at the design stage. We are particularly interested in assessing the impact of computer networking protocols on the control of manufacturing systems. In our design methodology, these systems are modeled using Petri nets. In this context, we propose an approach to modeling network protocols based on Oriented Object Petri Nets. Our ultimate objective is to assess by means of simulations the performances of such a system when one distributes their control models on an operational architecture. In this study, we are implementing a component based approach designed to encourage reuse when modeling new network protocols. To illustrate our approach and its reuse capabilities, we will implement it to model the link layer protocols of the norms IEEE 802.11b and IEEE 802.3.

1. Introduction

The development of computer networks has enabled the emergence of new applications benefiting from the power and flexibility offered by the distribution of their functions on different computers. We are interested particularly in the networked control of manufacturing systems which are a class of distributed discrete event systems. These systems, where flexibility and reconfiguration capacities are essential features, are increasingly dependent on computer networks.

The work presented in this paper is part of a broader approach on the design of distributed systems by the evaluation in the design phase of the impact of network protocols on the distribution on different computers of the functions of a distributed system.

Our actual methodology for the design of the control of manufacturing systems is based on the use of Petri nets. In order to have a uniform modeling tool allowing making assessments by the means of simulations, we have also opted for the use of Object-Oriented Petri Nets (OOPN hereafter) to model network protocols.

We propose a modeling approach that answers all the constraints of communication protocols. More particularly, we address constraints such as timing and synchronization that are important in distributed discrete event systems. We also take the stochastic requirement into consideration for the bit rate errors and the transmission depending on the services. Another constraint is to be able to analyze the impact of other traffics on a specific one between two workstations.

The main difficulties of the modeling phase of a system are generally due to its size and the precision of the model one wants to obtain. One of the methods to overcome such problem is the use of *generic and modular modeling* which consists of dividing the model into small reusable components and to deal with them *separately.* As an example, several LAN protocols own same features such as the backoff mechanism for medium access. Our approach proposes in addition generic reusable components for features such as backoff.

The paper is organized as it follows. Section 2 gives a mathematical definition of Object-Oriented Petri Nets. In Section 3, we focus on the modular modeling. Section 4 presents a component-based approach with goal to develop a library of reusable components based on Petri Nets formalism. In this section, the reusability of our components is illustrated by two examples: IEEE 802.11b DCF and Ethernet. In Section 5, we use simulation to prove the correctness of our models.

978-0-7695-3586-9/09 $25.00 © 2009 IEEE
DOI 10.1109/ICNS.2009.101

2. Object-Oriented Petri Nets

Petri nets have been proposed by C. A. Petri in 1962 [1, 2]. Petri nets are a powerful modeling formalism in computer science, system engineering and many other disciplines. They are used to study and describe different types of systems: distributed, parallel, and stochastic; mainly *discrete event systems*.

An Object-Oriented Petri Net [3, 4] *OOPN* can be considered as a special kind of high level Petri nets which allow representing and manipulating objects. In OOPN, tokens are considered as tuples of instances of object classes which are defined as lists of attributes. It can represent all parts of complex systems, increasing the flexibility of the model. It is a collection of elements comprising constants, variables, net elements, class elements…

Based on high level object oriented programming language, an OOPN system is composed of mutually communicating physical objects and their interconnection relations. From the mathematical point of view an OOPN is defined as: N = (O, W) where:

- O is a set of physical objects in the system, $\{O_i, i = 1, 2, \ldots, I\}$ (I = the total number of physical objects in the system).
- W is a set of message passing relations among distinct objects in the system $\{W_{i,j}, i, j = 1, 2, \ldots, I; i \neq j\}$. W is defined as: $W_{i,j} = (OP_i, GT_{i,j}, IP_j)$ where: $GT_{i,j}$ is a special transition called gate transition.

A physical object can be defined as $O_i = (P_i, T_i, IP_i, OP_i, IA_i, OA_i, M_i, \Sigma_i, G_i, \Lambda_i, E_i)$ where:

- P_i is a set of state places in Oi, $P=\{p_1, p_2, \ldots, p_m\}$,
- T_i is a set of active object transitions in Oi, $T=\{t_1, t_2, \ldots, t_n\}$,
- IP_i is a set of input message places,
- OP_i is a set output message places,
- IA_i is a set of the input transition arcs,
- OA_i is a set of the output transition arcs,
- M_i is the input and output relationships between active transitions and state/message places for the physical object O_i.
  $M \subseteq (P\times T) \cup (T\times P) \cup (IP\times T) \cup (T\times IP) \cup (OP\times T) \cup (T\times OP)$,
- Σ_i is a finite set of non-empty color sets in O_i,
- Λ_i is a color function in O_i, Λ: P, OP, IP $\rightarrow \Sigma$,
- G_i is a guard function in O_i, G: T $\rightarrow$ Boolean expression, where:
  $\forall t \in T$: $[\mathrm{Type}(G(t)) = B_{exp} \wedge \mathrm{Type}(\mathrm{Var}(G(t))) \subseteq \Sigma]$
- E_i is an arc expression function in O_i, E: IA, OA $\rightarrow$ E(a), where:
  $\forall a \in A$:$[\mathrm{Type}(E(a))=\Lambda(p(a)) \wedge \mathrm{Type}(\mathrm{Var}(E(a))) \subseteq \Sigma]$, (A is the input and output transition arcs)
  p(a) is the place of arc a.
- I_i is an initialization function in O_i, I: P $\rightarrow$ a closed expression I(p) (without variables) where:
  $\forall p \in P$: $[\mathrm{Type}(I(p)) = \Lambda(p)]$

An OOPN is taking into account all the aspects of network protocols (frame identification, temporal expressions, stochastic behavior …).

3. Modular Component-Based Modeling

A *modular model* [5] is made of independent elements viewed as *black boxes* connected together. The modification, adding or removing of an element in a model can be done easily. This composite structure can be found at different abstraction levels: a black box itself can be made of smaller blocks. This gives the possibility to treat its components in a separate stage. In addition, the use of small pieces helps in upgrading the model.

Another advantage, with blocks working alone, is the opportunity to *reuse* them to reduce the overall cost. Blocks libraries can be developed specifically to handle communication networks.

3.1 Modeling Communication Protocols

Since we use Petri nets (PN) to model the different component behaviours, the inputs of a module are places and the outputs are modeled by transitions. This choice is coherent with the traditional way to model asynchronous communication between processes modeled by Petri Nets. A producer module fires an output transition to put tokens in the input places of consumer modules. The workstation detects signals and checks channel changes

The modeling process is helpful in studying the performance analyses of communication protocols. However, this modeling process can also be used in a design stage to check the correctness of a protocol. In both cases, it is important that the model allows static and dynamic checking. Static checkings are for example analyses of PN properties such as boundness, liveness and non-blocking. By dynamic checking one wants to mean the capacity to simulate the model. So, modeling the architecture of protocols for discrete event systems will help to verify the quantitative and qualitative properties of these systems.

3.2 Connectors between Modules

In our approach, the input and output gates of a module are respectively modeled by places and transitions and connected by arcs. The connection between transitions and input places between two blocks can be 1-to-many, many-to-1 or 1-to-1. As an example, figure 3.1 shows a many-to-1 and a 1-to-many connections. A many-to-1connection is used to connect workstations output transitions to a medium input place since workstations put their data on the medium only. A 1-to-many connection is used to connect the medium output transitions to workstations input places, since all the workstations can see the signals propagating on the medium.

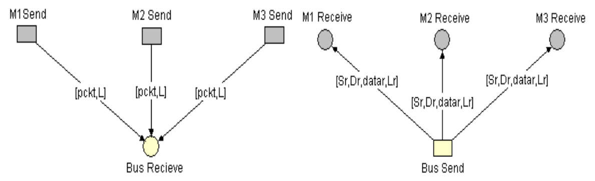

Figure 3.1, Many-to-1 and 1-to-many connection

This approach is very useful to deal with the complexity due to the size of a system. Indeed, if one has already a model of some stations connected on a bus and one wants to increase the size of its model, the connection of new workstations can be done easily just by adding an arc between the output transition of the bus model and the input place of the station model. Conversely, if the transitions are used as input gates and places as output gates, the addition of new workstation would need to add a new token in the output place.

As an example to model IEEE 802.11 protocol, one can consider two modules: workstation based module and medium based module. Figure 3.2 shows the inputs/outputs of workstation and medium modules. The workstation module is with one output transition “Send”, and three input places “Receive”, “Busy” and “Idle”. These gates represent the signals that a workstation can send or receive.

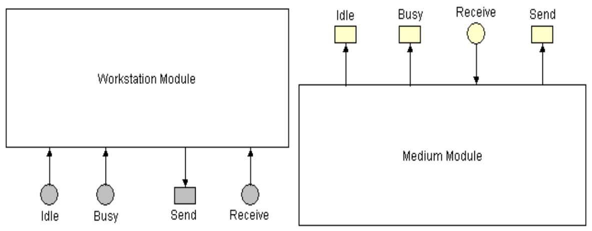

Figure 3.2, Workstation and Medium Modules

The medium module is with one input place “Receive”, and three output transitions “Send”, “Busy” and “Idle”. As we can see, these gates are the same as the gates on the workstation module with respect to directions.

4. Component-Based Application Models

In the rest of this paper, we will illustrate our modeling method through the examples of IEEE 802.11 MAC protocol [6] and Ethernet IEEE 802.3 MAC protocol. 802.11 protocol is a wireless MAC protocol, IEEE standard, for *Wireless Local Area Network WLAN*. It is widely used in the wireless mobile internet. In 802.11, there are two mechanisms to access the medium in a fair way. The basic mechanism is the *Distributed Coordination Function DCF* [7]. It is a random access technique based on the *carrier sense multiple accesses with collision avoidance (CSMA/CA)* mechanism. The second mechanism to access the medium in 802.11 is the *Point Coordination Function PCF* [8] or *Priority-based access* which is a centralized MAC protocol.

4.1 Channel Check

When a workstation wants to transmit over CSMA/CA, it must first sense if the channel is idle for more than a period of time called *Distributed Inter-Frame Space DIFS*. If so, it starts a random *backoff*. During the backoff time, it continues sensing the channel. If the channel stays free during the backoff, it can send its packet. However, if the channel becomes busy, it stops decrementing the backoff, but it keeps its remaining value (CA). Then, it repeats the first step. The remaining value of the backoff is decremented. Figure 4.1 shows the access method to the channel.

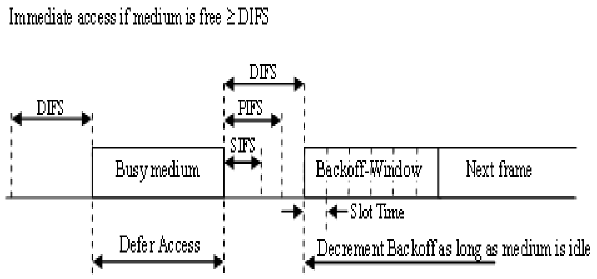
Immediate access if medium is free ≥ DIFS
DIFS
DIFS
PIFS
SIFS
Busy medium
Backoff-Window
Next frame
Slot Time
Defer Access
Decrement Backoff as long as medium is idle
Figure 4.1, DCF access to channel

Figure 4.2 shows the channel check component. The *dark* places and transitions represent the interfaces of this module, the *hashed* ones represent the reused places and transitions to build this component and the *white* ones represent the connected parts from other components. Initially the channel is free. This is represented by the presence of a token in place “Idle”.

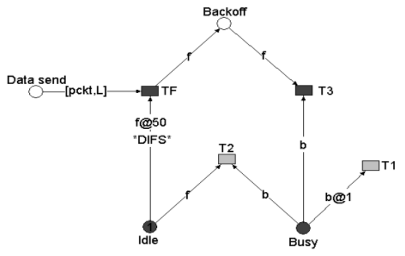

Figure 4.2, Channel Check

If the workstation wants to transmit data, a token is put in place “Data send”. If after 50µs the channel stays idle (the notation *@50* means that the token in place “Idle” must stay for 50 units of time after putting a token in place “Data send” before transition TF can fire), transition TF can be fired and the workstation can start its random backoff.

If the channel status is changed (a token is put in place “Busy”), the workstation can be in three cases: (i) it is during the DIFS waiting period, then T2 is fired, (ii) it is during the backoff period, then T3 is fired, (iii) it has no data or it is the transmitter to send then T1. In any case, the channel status becomes idle.

4.2 Contention Window

The value of the backoff depends on the *contention window CW* value. The workstation picks a number between zero and CW. The picked value is multiplied by the slot time to have backoff value. To decrement the backoff, the workstation continues checking the channel and each time the channel is free for a time slot, it decrements one of the picked value. However, if a collision occurs (detected by using a watchdog technique associated with the receipt of an ACK sent back by the recipient workstation) the value of CW is doubled. The minimum value of CW or *CWmin* equals to 16 and the maximum value or *CWmax* equals to 1024. Once a successful reception, the value of CW return to CWmin.

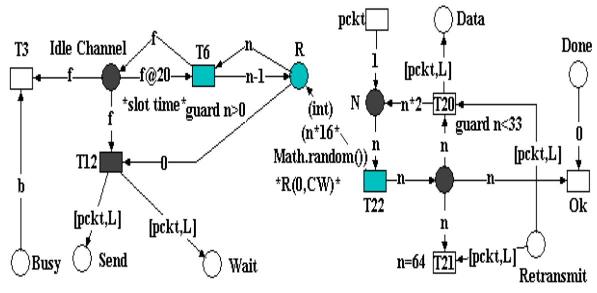

Figure 4.3, Backoff Decrementation with OOPN

Figure 4.3 proposes an OOPN modeling of the backoff mechanism. At the beginning, the token in place N (number of transmissions) is initialized to 1. The value of N is multiplied by 16 to determine the value of CWnew. The normally distributed function *Math.random()* is used to pick a random value between 0 and 15 (16 is not included, and the function “*int*” returns an integer value between 0 and n*16). The transition T6 has a guard that must be true to be enabled: the value of *n* must be greater than zero (the value of *n* is decremented each slot time when the channel is always idle). Once *n* equals to zero which is a condition on the arc, T12 is enabled.

Transitions T20 and T21 are used as a watchdog for the retransmission. So, if the maximum value of CW is reached then the packet cannot be retransmitted, and hence T21 is fired. Otherwise T20 is fired. After a successful transmission the counter N is reset to zero by firing transition “OK”.

4.3 Receiving Data and Sending ACK

Once the workstation sends its packet, it waits for a time equals to SIFS and checks if it receives an acknowledgement or not. If it does not receive an acknowledgement after SIFS (10µs in 802.11b), it doubles the backoff and restarts the transmission process.

Figure 4.4 shows the receiving process. The workstation has one receive link, so it receives both ACK and data packets. It checks first if the packet belongs to it or not. The guard condition associated with transition T15 checks if the received frame is for the considered workstation. Next the guard condition of transition T10 checks if the received packet is an ACK. If it is not an ACK frame, then the T11 is fired. Hence, T10 and T11 are never in conflict and T10 is not fireable if the workstation is not the transmitter because a token must be put in place “wait” since the firing of T12.

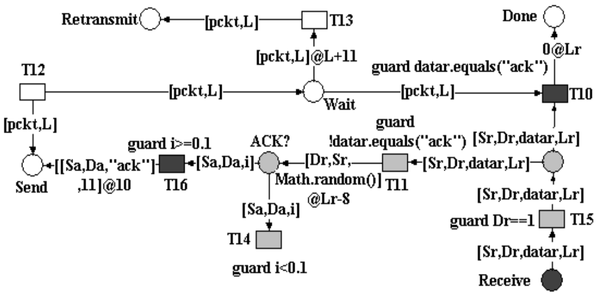

Figure 4.4, Receiving Data and Sending ACK

The transition T13 models a watchdog mechanism to check if the ACK is not received after a period depending on the length of the sent frame. L+11 represents the time needed to transmit the data frame plus a waiting time greater than SIFS. As in an OOPN, the token belongs to an object class, one can define as many n-tuple based on the token attributes. As an example, the arc between place “Transmission” and transition T12 is labeled by [S, D, data, L]. This n-tuple is useful to characterize the source address of the frame (attribute S), the address of the receiver (D), the data of the frame and also the transmission time of the frame that is equivalent to its length L. From figures 4.3 and 4.4, one can see that OOPN have a modeling power comparable with temporal, stochastic and colored Petri nets.

4.4 Component Composite Approach

Figure 4.5 shows a detailed OOPN of a wireless workstation, modeled with "*Renew 2.1*" [9]. All workstations have potentially a bandwidth of 11 Mbps (Without considering the bandwidth attenuation which depends on the distance between the two stations). However, distance attenuation or bandwidth drop can be modeled since we use a dynamic function to represent the packet transmission as a function of time.

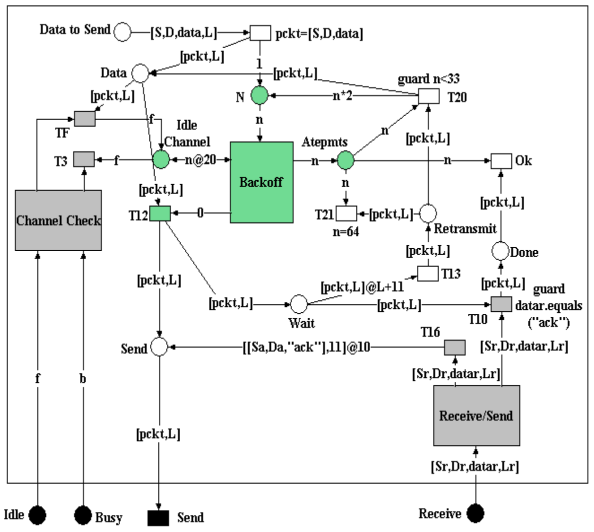

Figure 4.5, A detailed OOPN of a Wireless Workstation

The hashed blocks represent the reused components. The white places and transitions are used to complete the connections between these components to build the composite. While the black places and transitions represent the interfaces of the composite (seen as black-box).

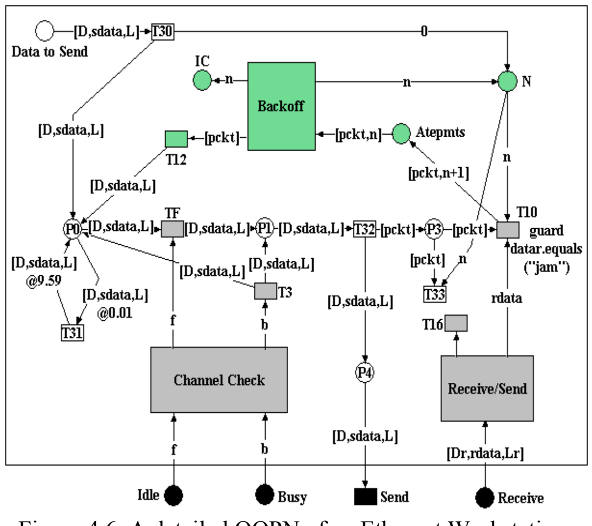

Figure 4.6, A detailed OOPN of an Ethernet Workstation

One of the benefits that one can get from modularity is the ability to reuse some components from previous models. If we take the Ethernet 802.3 protocol [10], the channel check, the reception of packets or the backoff processes are nearly the same (we just need to modify some values but not the structure of the components). This means that we can build a library of components that can be used in many other systems and save cost of modeling.

Figure 4.6 shows an Ethernet workstation module. As one can see, there are some differences between both modules that transition T16 is not connected to place "Send" since no acknowledgment is send over Ethernet network. Also place "IC" is not used since no channel check is needed in Ethernet. The parameters on the arcs are also not the same since they depend on the systems itself.

5. Simulation and Results Comparisons

To evaluate the quality and accuracy of our model, let us compare the obtained performances with data given by other studies about IEEE 802.11b network [12, 13] and also the results of NS-2 simulation performed in the same conditions. Our simulations are based on full-mesh dense networks with different numbers of workstations. The simulation can be performed step by step or continuously.

Each simulation assumes that all nodes transmit at 11Mbps and all nodes try to send data as soon as possible. Each host has 1000 packets with average length of 1150 bytes. Both simulations are achieved on Intel® Core™ 2 Duo Processor T2300.

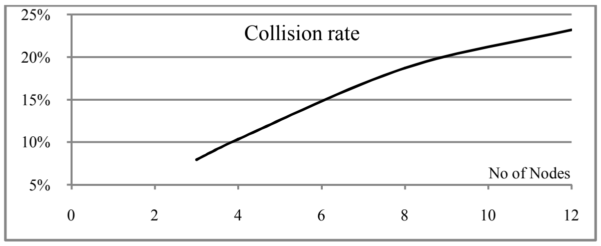

Figure 5.1, Collisions rate percentage

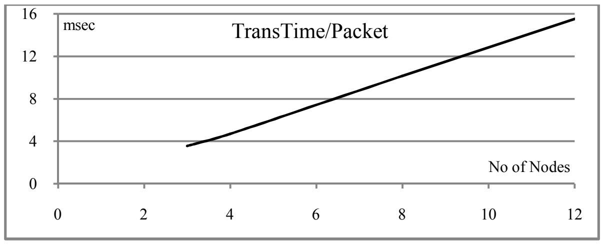

Figure 5.2, Time needed to transmit a packet in msec

Figure 5.1 shows how the collision rate increases when the number of workstations increases, while figure 5.2 shows the time needed to transmit one packet depending on the number of nodes on the network. The transmission time increases linearly due to the number of sent packets on the network.

Table 5.1 shows the collision rate the bandwidth per node and the total effective bandwidth. The last one is decreasing due to the collision rate increment.

Table 5.1, Collision rate, Total Bandwidth and Time / Packet

No of Nodes	Collision rate	BW/ Node Mbps	Time/ Packet ms	Total effective BW
3	7,95%	2.76	3.541	8,29 Mbps
4	10,34%	2,06	4.694	8,25 Mbps
8	18,70%	0,92	10.15	7,34 Mbps
12	23,18%	0,58	15.52	6,97 Mbps

Figure 5.3 shows the throughput of 802.11b nodes sharing a bandwidth of 11Mbps. As we can see, the values are nearly the same with our approach and with NS-2 based simulations. This result confirms the correctness of our model.

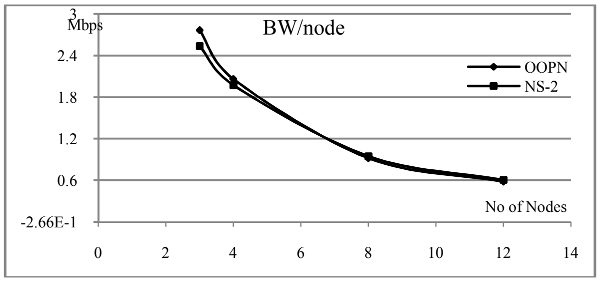

Figure 5.3, Bit Rate Variation with Number of Nodes

As one can see in figure 5.4, the simulation time increases in a linear way when the number of nodes is increased. If one returns to figure 5.2, one can see the confirmation of these results.

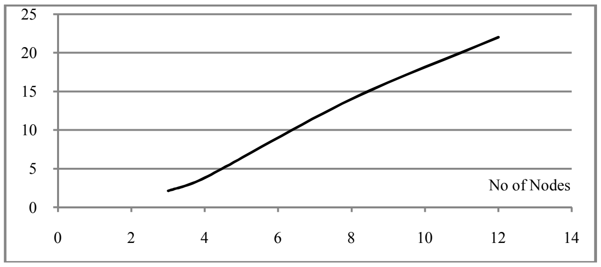

Figure 5.4, Simulation Time verses number of nodes

6. Conclusion

A component-based modular approach based on OOPN allows modeling in the same formalism network protocol and the services of a DES distributed application. As an example, we have chosen wireless protocol IEEE 802.11b and Ethernet 802.3. With means of simulation to evaluate our model, NS-2 and a comparison with existing measurements, we proved the correctness of our technique and the nearly exact results that we have obtained from the analysis. We have also examined the reusing of generic components which reduces the time and the cost needed to build new models.

In the future, we want to propose a complete modeling framework that covers also services and applications. This will allow a designer building a model depending on the user specifications, and just by selecting the most appropriate basic models in given libraries.

7. References

[1] T. Murata. "*Petri nets: Properties, Analysis and Applications.*" Proc. of the IEEE, 77(4), 1989.

[2] C. Law, B. Gwee, J. Chang "Fast and memory-efficient invariant computation of ordinary Petri nets." IET Computers and Digital Techniques, VOL 1(5), 2007.

[3] C. Lakos. "*From Coloured Petri Nets to Object Petri Nets.*" Lecture Notes in Computer Science, VOL 935, PATPN, 1995.

[4] Z. YU, Y. CAI. "*Object-Oriented Petri nets Based Architecture Description Language for Multi-agent Systems*", IJCSNS, VOL 6(1), 2006.

[5] C. Potts, "*A Generic Model for Representing Design Methods..*" 11th International Conference on Software Engineering, 1989.

[6] IEEE Computer Society. "*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*" IEEE Std. 802.11™-2007.

[7] G. Bianchi. "*Performance Analysis of the IEEE 802.11 Distributed Coordination Function.*" IEEE Journal on Selected Areas in Communications, Vol. 18, No. 3, March 2000.

[8] T. Suzuki, S. Tasaka. "*Performance evaluation of priority-based multimedia transmission with the PCF in an IEEE 802.11 standard wireless LAN.*" IEEE PIMRC 2007.

[9] http://www.renew.de/

[10] IEEE Std 802.3™ "*Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.*" 2002.

[11] http://www.isi.edu/nsnam/ns/index.html

[12] G. Anastasi, E. Borgia, M. Conti, E. Gregori. "*IEEE 802.11b Ad Hoc Networks: Performance Measurements.*" Cluster Computing VOL 8, 2005.

[13] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda. "*Performance anomaly of 802.11 b.*" INFOCOM, 2003.