Performance Analysis of IEEE 802.11b Wireless Networks with Object Oriented Petri Nets

Aladdin Masri[1] Thomas Bourdeaud'huy[2] Armand Toguyeni[3]

LAGIS - CNRS UMR 8146
Ecole Centrale de Lille - Cit Scientifique BP 48
59651 Villeneuve d'ASCQ, France

Abstract

Communication protocols are often investigated using simulation. This paper presents a performance study of the distributed coordination function of 802.11 networks. Firstly, our study illustrates the different classes of Petri Nets used for modeling network protocols and their robustness in modeling based on formal methods. Next we propose a detailed 802.11b model based on Object-oriented Petri Nets that precises backoff procedure and time synchronization. Then, performance analyses are evaluated by simulation for a dense wireless network and compared with other measurements approaches. Our main goal is to propose a modular model that will enable to evaluate the impact of network performances on the performances of distributed discrete event systems.

Keywords: LAN protocols modeling, Petri Nets, Performance Analysis, 802.11b Standard, Simulation.

1 Introduction

Wireless technology has become popular to access to the internet and communication networks. The IEEE 802.11 offers the possibility to assign part of the radio channel bandwidth to be used in wireless networks. IEEE 802.11 has two ways to access the channel: *Point Coordination Function PCF* and *Distributed Coordination Function DCF* that uses CSMA/CA which allows sharing the channel fairly based on best effort. The characteristic of wireless networks vary from the wired networks. The method to access the channel, in DCF mode, requires checking if the channel is idle for more than a *DIFS* (Distributed Inter Frame Space). Then, it begins its transmission after a random backoff based on the value of the contention window *CW*. It must receive an acknowledgment from the destination, after a *SIFS* (Short IFS) time, to guarantee a successful transmission, otherwise it will assume that the frame is in collision and retransmit it as above.

[1] Email: aladdin.masri@ec-lille.fr
[2] Email: thomas.bourdeaud_huy@ec-lille.fr
[3] Email: armand.toguyeni@ec-lille.fr

In this paper we propose an Object-Oriented Petri Nets modeling approach that is a brick to model the impact of networks' protocols on the performances of distributed discrete event systems. We develop a model that fulfills all the constraints of communication protocols. The main constraints are timing and synchronization of workstations especially for distributed systems. We also take the stochastic requirement into consideration for the bit rate errors and for the transmission depending on the services. Another constraint is the ability to analyze the impact of others traffics on a specific one between two workstations. The approach also proposes in addition the modeling of backoff, collision procedure and a dynamic length of data frames.

The paper is organized as follows. Section 2 gives a mathematical definition and a comparison of the different classes of Petri Nets. We discuss the benefits and weakness points for each class in the modeling of communication protocols. Section 3 gives a brief introduction to IEEE 802.11b DCF and presents our model. At the end, performance analysis is validated by means of simulation.

2 Petri Nets For Modeling Network Protocols

Petri nets have been proposed by C. A. Petri in 1962 [1]. 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.* Petri nets are in two forms: *mathematical and graphical.*

2.1 Modeling with Ordinary Petri Nets

An ordinary Petri net $\boldsymbol{N=(P,\ T,\ A,\ m_0)}$ can be defined as a bipartite directed graph, where:

- **P** and **T** are the sets of nodes respectively called places and transitions ($|P| = m, |T| = n$);
- **A: P**$\times$**T** $\cup$ **T**$\times$**P** $\rightarrow$ **N** is the weighted flow relation representing the arcs;
- $\mathbf{m_0}$**:** **P** $\rightarrow$ **N** is a mapping associating to each place p$\in$**P**, an integer $m_0(p)$ called the initial marking of the place p.

The marking of a Petri net can be modified by the firing of transitions. A transition t is fireable from a marking m_a (denoted by $m_a[t\rangle$), when $\forall$p$\in{}^o\mathbf{t}$ with ${}^o\mathbf{t}$={p$\in$**P** such as **A**(p, t)>0}, $m_a(p) \geq \boldsymbol{A}(p,\ t)$. If this condition is satisfied, a new marking m_k is produced from the marking m_a(denoted by $m_a[t\rangle m_k$): $\forall$p$\in$**P**, m_k(p)= m_a(p) + **A**(p, t) + **A**(t, p).

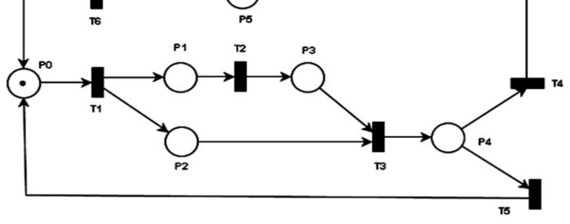

Fig. 2.1 Example of an Ordinary Petri Net

Fig. 2.1 shows an example of an ordinary Petri net. In this figure one can see some characteristics of a Petri net. The transition T2 cannot be fired before the firing of T1. This characteristic is called the *sequential execution.* T3 is a *synchronisation transition* since it is enabled as soon as P2 and P3 have tokens. Transition T4 and T5 are in *conflict* since only one of them can be fired when P4 receives a token. However, there are some problems to model computer protocols with ordinary Petri nets:

(i) *Time modeling.* Ordinary Petri nets do not handle time. This makes it difficult or even impossible to model communication protocols with such Petri nets because time is one of the main features of network protocols.

(ii) *Priority and stochastic modeling.* These characteristics do not exist in ordinary Petri nets. This does not solve the conflict problem or cannot define a probability to fire such transitions.

2.2 Modeling with Timed Petri Nets

Timed Petri nets are a class of Petri nets. It was introduced by C. Ramchandani in 1974 [2]. It is seen as $\boldsymbol{N= (P,\ T,\ A,\ m_0,\ \tau)}$ where $\boldsymbol{(P,\ T,\ A,\ m_0)}$ is an ordinary Petri net, and τ: $\boldsymbol{T \rightarrow R^+}$ is a function that associates time delays to transitions.

In a timed Petri net, it is not necessary that a clock is associated to every transition. However, the time delays associated to the transitions modify the *marking validity conditions.* When a transition is fired, the token(s) in the input place(s) seems as it *disappears* and then it *reappears* after a period equals to delay associated to that transition. As a result, the beginning and end moments of transitions firing play a fundamental rule in the behavior of the timed Petri net which means one must take care of the delays desired to fire transitions.

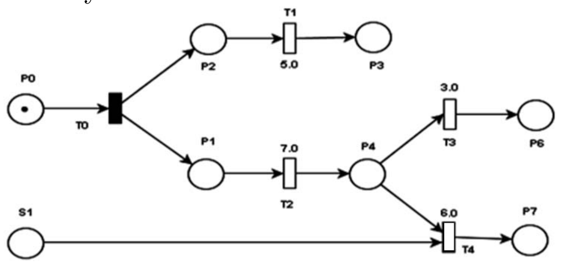

Fig 2.2 A Timed Petri Net

One can see in Fig. 2.2 that T0 is an immediate transition while T1, T2, T3 and T4 are timed transitions. So, after firing T0, both T1 and T2 are enabled. T1 can be fired after 5 units of time, and T2 can be fired after 7 units of time. T3 is now enabled but it has to wait 3 units of time before firing. Suppose that S1 (another workstation sending data) had a token before, T4 is now enabled. The choice of T3 or T4 determines to which place the token in P4 will go and after what time.

This may help in solving the conflict problem since both transitions are controlled by time, but still not for immediate transitions. However, this does not solve the stochastic problem. Suppose 90% of the packets of P4 may go to P6 and 10% may go to P7 (the bit rate error modeling in communication protocols), the timed Petri nets does not answer this characteristics. Another problem that one cannot model with this class is the time variation or intervals. As the time delays associated to transitions is constant, the occurrence possibility in an interval of time cannot be modeled. As an example, the length of a packet sent on a network varies from 60 to 1514 bytes. The time needed to send such packets depends on the length of that packet. With constant timing values, this action cannot be modeled easily or one must complexify the model.

2.3 Modeling with Time Petri Nets

Time Petri nets [3] [4] is a more powerful formalism used to model systems where time is the main constraint such as communication protocols and real-time systems. A TPN is a five-tuple $\boldsymbol{N=(P,\ T,\ A,\ m_0,\ \tau_s)}$ where τ_s**:** $\boldsymbol{T} \rightarrow \boldsymbol{R}^+ \times \boldsymbol{R}^+ \cup \{+\infty\}$ is a function called *Static Interval function*. Time is represented in intervals with lower min and upper max limits which make it easy to model events with unknown occurring time. The two limits min and max (with $0 \leq \min \leq \max$, $\min \in \mathbf{R}^+$ and $\max \in \mathbf{R}^+ \cup \{+\infty\}$) are associated to each transition. These limits are related to the date when $\mathbf{t}_i$ was enabled for the last time. Let θ be the date when $\mathbf{t}_i$ becomes enabled; then $\mathbf{t}_i$ cannot be fired before $\theta+min$ and must fire no later than $\theta+max$ (if max is finite), except if the fire of another transition $\mathbf{t}_j$ un-enables $\mathbf{t}_i$ before it is fired. Transition firings have no durations.

The transition firing in a Time Petri Net has two firing semantics. The first semantics is called the *strong firing semantics*, which impose that any enabled transition must be fired at its latest firing time at most. On the contrary, when using the *weak firing semantics*, the firing time of a transition is not constrained by firing conditions over other transitions. In this paper, we will use the strong firing semantics for the watchdog needs. In Fig. 2.3, in strong firing semantics, the transition T1 cannot be fired after 9 units of time since T0 must be fired before 9 units of time. T2 is an immediate transition.

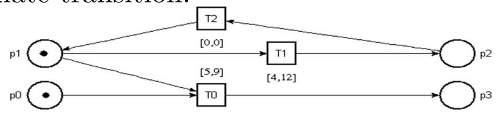

Fig. 2.3 A Time Petri Net

In some new tools, the TPN was improved with priority selection. In Fig. 2.4, transition T1 is fireable between 3 and 11 units of time. However, T0 has priority over T1, so T1 can be fired between 3 and 5 units of time but not later since T0 has priority at that time.

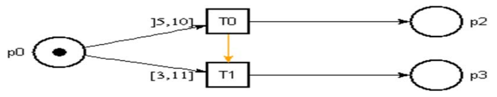

Fig. 2.4 Priority in TPN

Problems are not still all solved. The priority may be now solved but what about modeling complexity, percentage distribution or data addressing. In wireless networks, a workstation trying to send a packet must wait for a backoff delay before sending its packet. The backoff value is a random value between 0 and CW multiplied by the slot time. Fig. 2.5 shows how to model such action. This is just for random number, but what about data addressing or percentage distribution? TPN does not answer these questions now since it has no token identification or probability functions. In addition, if one insists, the complexity of the model prevents any analysis or what is known as the combinatorial explosion problem. For each workstation, the value of CW is at first 16, but after each collision (no acknowledgement received) the current CW value is multiplied by two, until it reaches 1024. So if one tries to get the state classes, it would be impossible since one have this huge number of tokens in just one place.

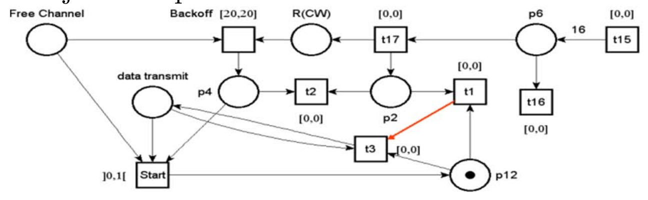

Fig 2.5 Random Backoff modeled with TPN

2.4 *Modeling with Stochastic Petri Net*

Stochastic Petri Nets [5] were proposed to integrate formal description, proof of correctness, and performance evaluation. They are Petri nets in which stochastic firing times are associated with transitions.

A Stochastic Petri Net is a tuple $\boldsymbol{N=(P,\ T,\ A,\ m_0,\ \Gamma)}$ where $\Gamma\boldsymbol{:\ T\rightarrow pdf}$ is a set of firing rates, and *pdf* is the *probability density function.* The entry $\delta_i \in \Gamma$ is an exponential distributed random variable, whose pdf is a negative exponential, associated with transition $\mathbf{t}_i$. The firing rate of any transition $\mathbf{t}_i$ may be marking-dependent, so it is necessary to be written as $\delta_i(\mathrm{M}_j)$. Thus, the average firing delay of transition $\mathbf{t}_i$ in marking M_j is $[\delta_i(\mathrm{M}_j)]^{-1}$. Since the rate is marking-dependent, when entering a marking, the transition with the minimum firing delay will be fired. Knowing that all the firing delays have *exponential pdf*, this allows saying that the probability for a given transition $\mathbf{t}_i$ with the minimum delay as:

$$\mathrm{P}(\mathrm{t}_i,\mathrm{M}_j)=\frac{\delta_i(M_j)}{\sum_{k:t_k\in X(M_j)}\delta_k(M_j)}$$

Where $\mathrm{X}(\mathrm{M}_j)$ is the set of all enabled transitions in the marking M_j.

So suppose there are three enabled transitions with firing rate $\alpha 1, \alpha 2, \alpha 3$ with minimum delay for $\mathbf{t}_1$, and then the probability of firing $\mathbf{t}_1$ is:

$$\mathrm{P}(\mathrm{t}_1)=\frac{\alpha 1}{\alpha 1+\alpha 2+\alpha 3}$$

The Generalized SPN, Fig. 2.6, is a subclass of the stochastic Petri nets, which allows immediate transitions in the net (which is not the case for SPN). A priority zero is given to timed transition while the immediate transitions own a priority higher or equal to 1.

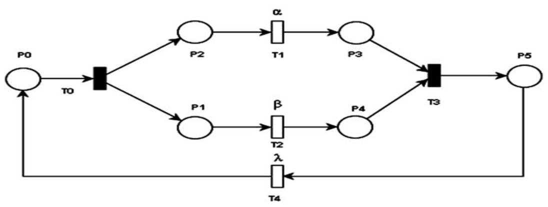

Fig. 2.6 A Generalized SPN

Stochastic Petri nets can now answer some problems and find solutions to them. But still not all the modeling problems are solved yet. Consider the communication between different workstations on the net, a workstation trying to communicate with another workstation must give the destination address so that the other workstations either forward the request or if it is the destination it will pick it up and stop forwarding the packet. In nearly all situations, the destination workstation sends an acknowledgement to the source workstation informing the reception of the message, otherwise it will repeat the transmission, as the wireless protocols [6] for example.

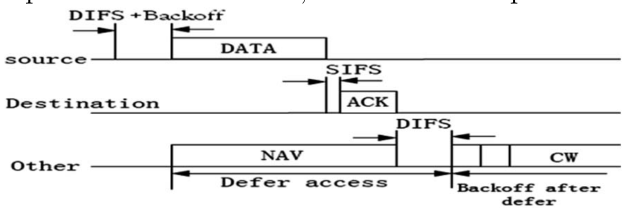

Fig. 2.7 Wireless Protocol message exchange process

This process needs to "label" the token with the name of the source, destination and the message an in Fig. 2.7. The stochastic Petri nets do not have the capacity to do this since tokens are all of the same type and have no modifiers.

2.5 Modeling with Colored Petri Nets

Colored Petri nets [7] have different characteristics from other classes, where token(s) and places are attached with a color identifying the type of that token and place. A CPN is a tuple $\boldsymbol{N=(P, T, A, m_0, \Sigma, \Lambda, G, E, I)}$ where:

- Σ is a finite set of non-empty color sets.
- Λ is a color function, Λ: P $\rightarrow \Sigma$.
- **G** is a guard function, G: T $\rightarrow$ Boolean expression, where:
  $\forall$t$\in$T: [Type(G(t)) = $B_{exp} \wedge$ Type(Var(G(t))) $\subseteq \Sigma$]
- **E** is an arc expression function, E: A $\rightarrow$ E(a), where:
  $\forall$a$\in$A: [Type(E(a))= Λ(p(a))$\wedge$Type(Var(E(a)))$\subseteq\Sigma$]], p(a) is the place of arc **a**.
- **I** is an initialization function, I: P$\rightarrow$**a** closed expression I(p) (without variables) where: $\forall$p$\in$P: [Type(I(p)) = Λ(p)]

From the above definition one can say that the color function defines the type (called multi-set type) of values in each place. Arcs' inscriptions must be a non-empty expression type that matches the color of the place to where it is connected (to fire a transition). The initial marking m_0 is obtained by evaluating the initialization expressions: $\forall$p$\in$P: m_0(p)= I(p) where m_0(p)$\in\Lambda$(p).

The firing of a transition in a CPN must satisfy some conditions:

(i) Input places of a transition $\mathbf{t}_i$ must contain the number of tokens enabling that transition: $\forall$p$\in{}^o\mathbf{t}$ with ${}^o\mathbf{t}$= $\forall$p$\in\mathbf{P}$ such as **A**(p,t)>0, m_j(p) $\geq$ **A**(p, t) and Type(E(a))= Λ(p(a))

(ii) The guard function associated with that transition must be true to enable the transition: G(t_i) = True

(iii) The output tokens (tokens in output places) submit to the output arc's inscription (color and number). Note, in some tools this inscription can be an empty function if condition is not satisfied; i.e. no tokens is produced $\{\phi\}$.

(iv) The new marking is defined as: m_k = m_j - E(A(p, t)) + (E(A(t, p)).

From the previous definitions one can see the modeling power of such formalism. The idea of defining tokens as color sets or structures means that the token is now identified since it contains data allowing differentiating it from the other tokens and it is just as any other one.

Fig 2.8 shows a simple CPN [8]. All the places are of the same color type *INT*. Places **R** and **S1** each contain one token of the same type. The transition T1 is fireable after 50 units of time and has no guard function, while T2 is immediate and has a guard function: token **m** coming from place **S2** must be grater than 10. Since **S2** contains no tokens, a token with the value of 5 is put in place O1 after 50 units of time.

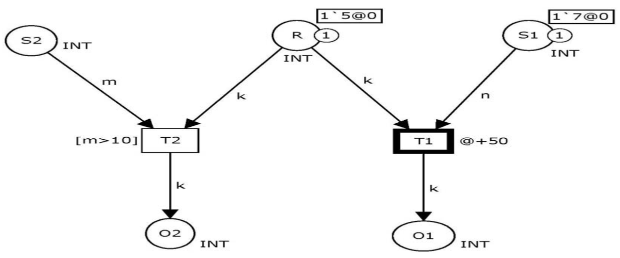

Fig. 2.8 A CPN example

Many works [9] [10] [11] were done on this formalism. However some of existing tools fall in simulation phase when the time is a main constraint. Returning to Fig. 2.8, if a token is put in place **S2** with a value greater than 10 and before the firing of T1 (during the 50 units of time), normally as the TPN definition, the transition must be fired since it is immediate. However, this is not always correct with such tool. During the simulation, both transitions can be fired which is not conform to the TPN definition.

2.6 Modeling with Object-Oriented Petri Nets

Not far from the colored Petri nets, the object-oriented Petri nets [12] [13] *OOPN* can be considered as a special kind of high level Petri nets which allow the representation and manipulation of 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, classes, object identifiers, and method net instance identifiers.

Based on high level object oriented programming language mainly Java or C++, OOPN takes all the meanings of object programming and the characteristics of Petri nets. From this perspective, an OOPN system is composed of mutually communicating physical objects and their interconnection relations. From mathematical point of view an OOPN is defined as: ***N=(O, W)*** where:

- **O** is a set of physical objects in the system.
- **W** is a set of message passing relations among distinct objects in the system.

A physical object can be defined as $\boldsymbol{Oi=(P_i, T_i, A_i, M_i, \Sigma_i, G_i, \Lambda_i, E_i)}$ where M_i the input and output relationships between transitions and places for the physical object O_i. From the above definition one can find the direct relationship between the colored Petri nets and object-oriented Petri nets. If one tries to look at the OOPN we find that nearly all the characteristics of Petri nets classes are in it:

(i) Since it is a Petri net, then it inherits the ordinary Petri nets.

(ii) The timing of a transition is as the definition of a TPN.

(iii) The use of a high-level programming language enforces it with all the mathematical function found in that language, especially when talking about stochastic and random expressions.

(iv) The structured tokens makes easier the modeling of complex systems like the discrete-event systems and communication protocols.

In section 3, we will illustrate different use examples of OOPN through protocols' modules modeled with OOPN.

3 Protocol Modeling with OOPN

802.11 [14] 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* [15]. 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* [16] or *Priority-based access* which is a centralized MAC protocol.

When a workstation wants to transmit over 802.11 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. Then, it repeats the first step in sensing the channel to be free for more than DIFS. The last value of the backoff is restarted and decremented. Fig. 3.1 shows the access method to the channel.

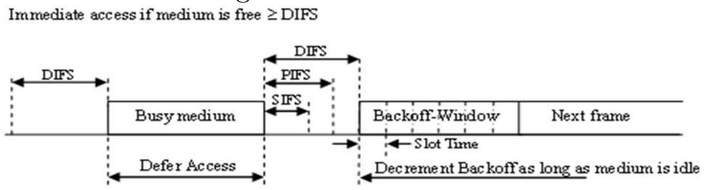
Immediate access if medium is free ≥ DIFS
DIFS
DIFS
PIFS
SIFS
Busy medium
Backoff-Window
Next frame
Slot Time
Defer Access
Decrement Backoff as long as medium is idle

Fig. 3.1 DCF access to channel

In 802.11b, a slot time equals to 20μs. *SIFS* or *Short Inter-Frame Space* equals to 10μs, and DIFS = SIFS + 2 * slot time = 10μs + 2*20μs = 50μs.

3.1 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 the backoff. 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 CW$_{min}$ equals to 16 and the maximum value or CW$_{max}$ equals to 1024. Once a successful reception is done, the value of CW returns to CWmin.

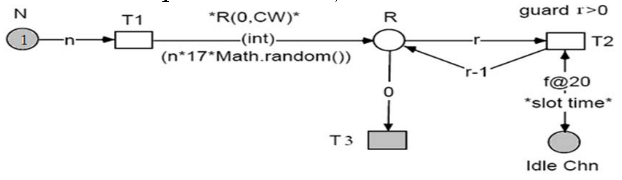

Fig. 3.2 Backoff Decrementation with OOPN

Fig. 3.2 proposes an OOPN modeling of the backoff mechanism. At the beginning the token in place N (number of transmissions) is initialized to 1. In case of collision its value is doubled. The value of N is multiplied by 16 to determine the value of CW$_{new}$. The normally distributed function *Math.random()* is used to pick a real random value between 0 and n*17 (17 is not included, and the function "int" returns an integer value between 0 and n*16). The transition T1 and T3 have no guards but T2 has a guard that must be true to be enabled which is the value of **r** must be greater than zero (the value of **r** is decremented each slot of time and the channel is always idle). Once **r** equals to zero which is a condition on the arc, T3 is enabled.

3.2 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 or 10μs, it doubles the backoff and restarts the transmission process. Fig 3.3 shows the receiving process. Since the workstation has one receive antenna, the workstation 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 "ACK?" from the firing of T12.

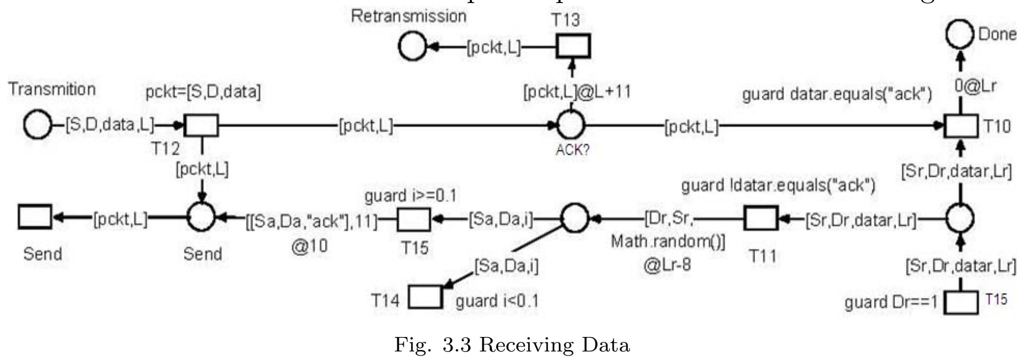

Fig. 3.3 Receiving Data

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 and a wait 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 3.2 and 3.3, one can see that OOPN have a modeling power comparable with TPN, SPN and CPN together.

Our approach considers two basic modules to model IEEE 802.11 network: workstation based module and medium based module. Fig.3.4 shows a detailed OOPN of a wireless workstation, modeled with "*Renew 2.1*" [17], and Fig. 3.5 for wireless medium. To design the medium module, one assumes that all workstations as potentially a bandwidth of 11 Mbps (Without considering the bandwidth attenuation which depends on the distance between two stations). The gray places and transitions in Fig. 3.5 are part of the workstations connected to the medium.

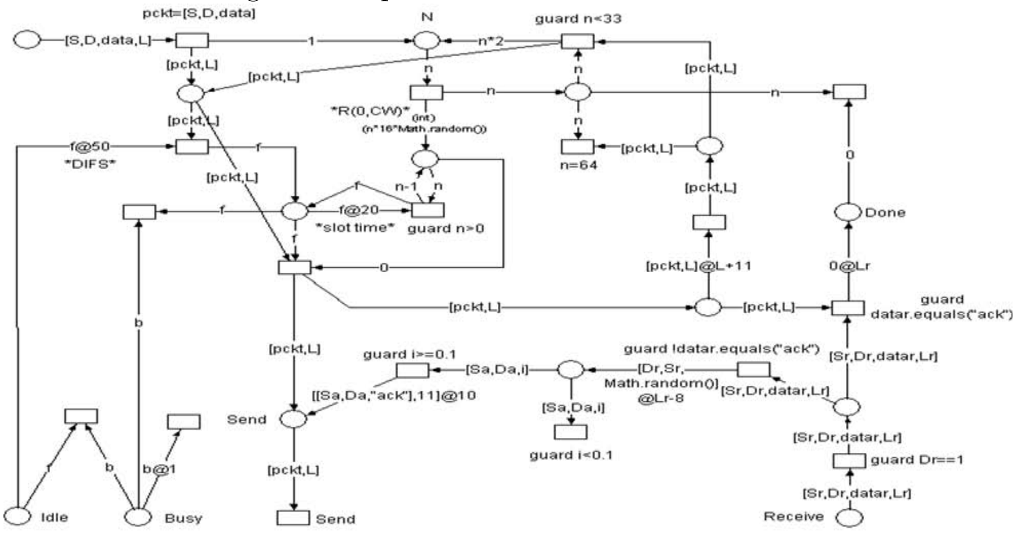

Fig. 3.4 A detailed OOPN of a Wireless Workstation

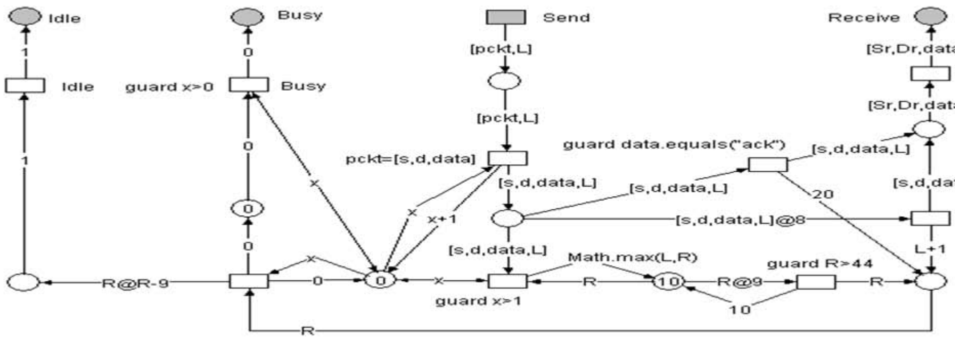

Fig. 3.5 A detailed OOPN of a Wireless Medium

3.3 Simulation and Results

Let us recall, our main objective in this study is to build protocols' bricks to be able to evaluate DES distributed architecture. So our first goal here is to evaluate the correctness of our models of IEEE 802.11b protocols. To achieve this evaluation, we have done different simulations of adhoc architectures. The obtained results were first compared with NS2 simulations' results that we have done, Fig. 3.7. We have also compared our results with others studies' results about 802.11b adhoc architectures, see [18] and [19]. We have verified that we obtain the same results. This proves the correctness and the quality of our OOPN modeling.

Our simulations are based on dense networks with different numbers of workstations. The 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.

Table 3.1 show the simulation results:

No of Nodes	Collision rate	BW/Node	Time/Packet	Total effective BW
3	7,95%	2.76 Mbps	3.541 ms	8,28750665
4	10,34%	2,06 Mbps	4.694 ms	8,24964632
8	18,70%	0,918 Mbps	10.153 ms	7,34179249
12	23,18%	0,58 Mbps	15.52 ms	6,96720672

Table 3.1 Collision rate, Total Bandwidth and time per packet

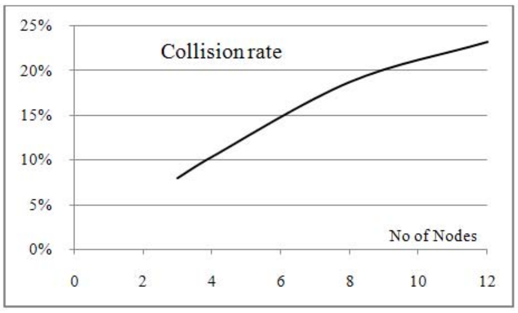

Fig 3.6 (a) collisions rate percentage

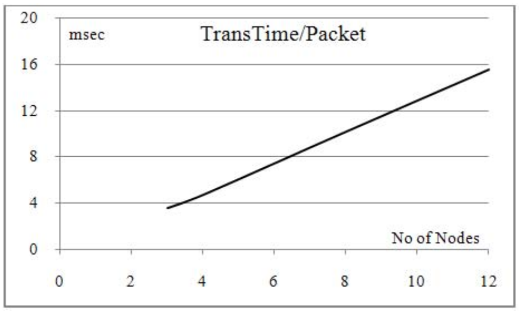

Fig 3.6 (b) time needed to transmit a packet in msec

Fig 3.6(a) shows how the collision rate increases when the number of workstations increases, while Fig. 3.6(b) shows the time needed to transmit one packet depending on the nodes on the network. Fig. 3.7 shows the throughput of 802.11b nodes sharing the 11Mbps.

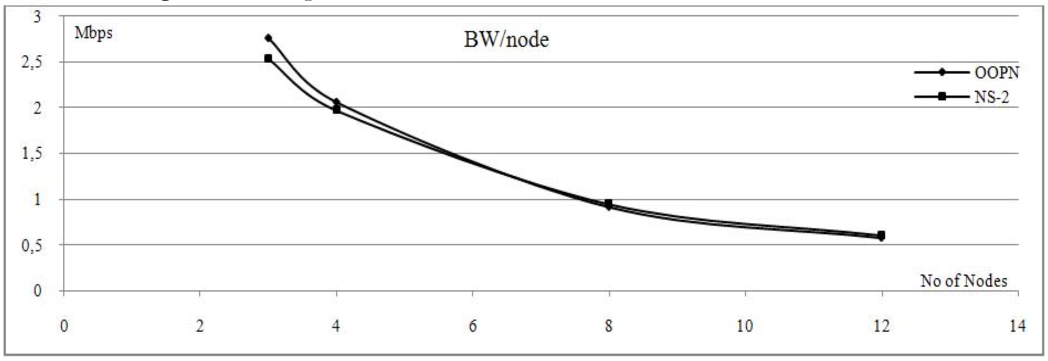

Fig. 3.7 Bit rate variation with number of nodes

4 Conclusion

In this paper we have proposed a modular OOPN approach that allows modeling in the same formalism a network protocol and the services of a DES distributed application in the future work. Let us recall here that our final goal is to be able to analyze the impact of network performances on a distributed application.

In this paper we have proved that Object-Oriented Petri Nets are well adapted to deal with all the constraints that must verify the model particularly with the possibilities to model stochastic or temporal behaviors and also to identify specific traffic. In this study, we have illustrated the capability of our approach by the simulations of IEEE 802.11b protocol and the comparisons of our results that are very closed to the values given by other studies. The modular feature of our approach allows proposing different models of same part of a system depending on the user requirement. As an example, we have shown that the medium model given here can be refined to consider the relative position of the different communicating stations. In the future, we want to propose a complete modeling framework that will allow a designer to build a model depending on the user specifications, and just by selecting the most appropriate basic models in given libraries.

References

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

[2] C. Ramchandani. "*Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.*" Project MAC, TR120, M.I.T., 1974.

[3] P. Merlin and D. Farber. "*Recoverability of communication protocols: Implications of a theoretical study.*" IEEE Tr. Comm., VOL 24(9), 1976.

[4] B. Berthomieu, F. Peres, F. Vernadat. "*Model-checking Bounded Prioritized Time Petri Nets.*" ATVA 2007. Springer Verlag, LNCS 4762, 2007.

[5] S. Natkin. "*Les Rseaux de Petri Stochastiques et Leur Application l'valuation des Systmes Informatiques.*" PhD thesis, Cnam, France, 1980.

[6] X. Wang, L. Wang. "*WLAN System Performance Evaluate Based on SPN.*" IEEE, ICCA, 2007.

[7] K. Jensen. "*Colored Petri Nets and the Invariant-Method.*" Theoretical Computer Science, Vol. 14, 1981.

[8] http://wiki.daimi.au.dk/cpntools/_home.wiki.

[9] R. Kodikara, S. Ling, A. Zaslavsky. "*Evaluating Cross-layer Context Exchange in Mobile Ad-hoc Networks with Colored Petri Nets.*" IEEE, ICPS, 2007.

[10] W. Mata, A. Gonz?lez, R. Aquino, A. Crespo, I. Ripoll, M. Capel. "*A Wireless Networked Embedded System with a New Real-Time Kernel - PaRTiKle.*" IEEE, CERMA, 2007.

[11] L. Liu, J. Billington. "*Verification of the Capability Exchange Signalling protocol.*" STTT,VOL p (3), 2007.

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

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

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

[15] G. Bianchi. "*Performance Analysis of the IEEE 802.11 Distributed Coordination Function.*" IEEE Journal on Selected Areas in Communications, VOL 18(3), 2000.

[16] 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.

[17] http://www.informatik.uni-hamburg.de/TGI/renew/.

[18] G. Anastasi, E. Borgia, M. Conti, E. Gregori. "*IEEE 802.11b Ad Hoc Networks: Performance Measurements.*" Cluster Computing VOL 8(2-3), 2005.

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