A Component-Based Approach Based on High-Level Petri Nets for Modeling Distributed Control Systems

Aladdin Masri
Department of Computer Engineering
An-Najah National University
Nablus, Palestine
e-mail: masri@najah.edu

Thomas Bourdeaud'hui Armand Toguyeni
LAGIS – CNRS UMR 8146
Ecole Centrale de Lille
59651 Villeneuve d'ASCQ, France
e-mail: thomas.bourdeaud_huy@ec-lille.fr,
armand.toguyeni@ec-lille.fr

***Abstract*—The evaluation of using *distributed systems DS* in place of centralized systems has introduced the distribution of many services and applications over the network. However, this distribution has produced some problems such as the impacts of underlying networking protocols over the distributed applications and the control of the resources. In this paper we are interested particularly in manufacturing systems. Manufacturing systems are a class of distributed discrete event systems. These systems use the local industrial network to control the system. Several approaches are proposed to model such systems. However, most of these approaches are centralized models that do not take into account the underlying network. In this context, we propose the modeling of the distributed services and the underlying protocols with High-Level Petri Nets. Since the model is large, complex and therefore difficult to modify, we propose a component-based modeling approach. This approach allows the reuse of ready-to-use components in new models, which reduces the cost of development. 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.**

Keywords-communication protocols; distributed systems; modeling; component-based; Petri nets;

I. INTRODUCTION

Distributed systems [1] [2] are increasing with the development of networks. 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 this work on the networked control of *manufacturing systems*. Manufacturing systems are a class of *discrete event systems* whose elements are interacting together to build products or to perform services. The concept of *flexible manufacturing systems FMS* has been introduced to develop new manufacturing systems able to produce small or average series of products.

Modeling such systems is very important to verify some properties, especially performance issues. In the literature, many models have been proposed to model manufacturing systems [3] [4] [5]. However, the classical modeling paradigm is generally based on a centralized point of view. Indeed, this kind of modeling does not take into account the fact that the system will be distributed when implemented over different machines, sensors, actors, etc. So, the properties obtained at the design stage are not necessarily guaranteed at the implementation stage.

In addition, the proposed models do not take into account the underlying network and protocols in terms of performance and information exchange. The behavior and design of manufacturing systems are affected by the underlying network features: performance, mobility, availability and quality of service characteristics.

A way to overcome such problems is to model these systems in a distributed way. A distributed system-model offers means to describe precisely all interesting forms of unpredictability as they occur. It takes into account each part of the system, available resources, and system changes together with the underlying network. Once this model is made, its implementation is easier since it has the same characteristic as the desired system. Nevertheless, these systems are complex: they show massive distribution, high dynamics, and high heterogeneity. Therefore, it is necessary to model these systems in a way that provides higher degree of confidence and rigorous solutions.

To cope with this challenge, we propose the use of a *component-based methodology* which is consistent with the principle of distributed systems in which elements are reusable and composable units of code. The component-based approach uses generic, hierarchical and modular means to design and analyze systems. It shows that the system model can be assembled from components working together and the designer needs only to identify the good components that offer suitable services with regard to applications requirements. This methodology allows the reusability and genericity of the components which reduces the cost of the systems development.

In this paper, we propose to model these systems with High-Level Petri Nets which is a powerful tool particularly dedicated to concurrent and distributed formalism, allowing to model both protocol and service components. The work presented in this paper is part of a larger approach on the design of distributed systems by the evaluation, in the design phase, of the impact of network protocols on the distribution of the functions of a distributed system on different computers [6] [7] [8].

The paper is organized as follows. Section 2 introduces the networked control systems and the communication architecture. In Section 3, we focus on the Petri nets modeling formalism. Section 4 gives the properties of modeled components: genericity, modularity, reusability and abstraction level (hidden implementation with connection interfaces). Section 5 shows our methodologies to build the basic components: a top-down analysis methodology and a bottom-up construction methodology. This section presents the component-based approach to develop a library of reusable components based on Petri Nets formalism. At the end of this section, we illustrate the reusability of our components by two examples: IEEE 802.11b DCF and Ethernet. In Section 6, we validate our modeled components by means of simulation to test the correctness of our models. Section 7 presents a case study for modeling manufacturing systems. In this section we show the impact of the underlying network protocols on the services offered by the system.

II. NETWORKED CONTROL SYSTEMS

Modern advances in hardware technologies and particularly in communication networks have played a big role in the rapid development of communication networks and distributed control systems.

A. Networked Control Systems Overview

Manufacturing systems are a class of discrete event systems. A Discrete Event System (DES) [9] [10] is a discrete-state, event-driven, dynamic system. The state evolution of DES depends completely on the occurrence of asynchronous discrete events over time. Fig. 1 shows the state jumps in a DES from one discrete value to another whenever an event occurs during the time. Nearly all the DESs are complex and require a high degree of correctness. Information systems, networking protocols, banking systems, and manufacturing and production systems fall into this classification.

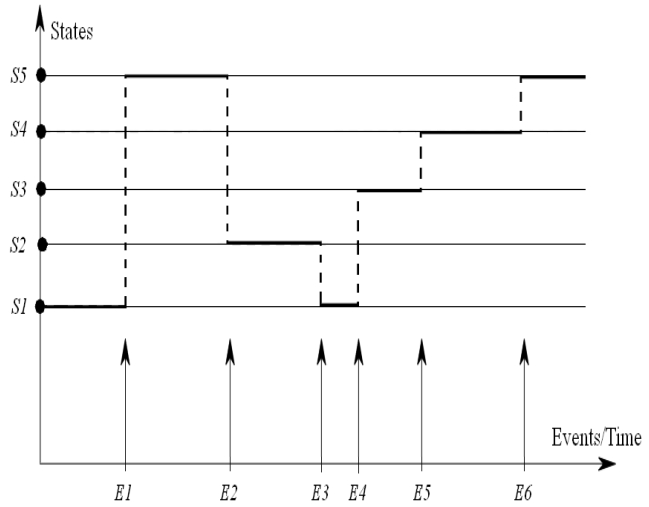

Figure 1. Discrete Event System

In order to improve the adaptability to the market and the quality of manufactured products and to allow their fast evolution, the implementation of *flexible manufacturing cells* is necessary. A flexible manufacturing system (FMS) is a production system that consists of a set of machines connected together via an automatic transportation system. Machines and transportation components such as *robots* are controlled by numerical controllers. In all cases, additional *computers or programmable logical controllers PLC* are used to coordinate the resources of the system.

The cell controllers or computers have a lot of functions and are used to control all the operations of an FMS. The control system manages most of the activities within an FMS like parts transportation, synchronising the connection between machine and transportation system, issuing commands to each machine, etc. *Networking* is extensively applied in industrial applications. The connection of the system elements through a network reduces the system complexity and the resources cost. Moreover, it allows sharing the data efficiently.

Thus, the control of such systems is very important. Nowadays, a controlled system [11] [12] is the combination of sensors, actuators, controllers and other elements distributed around a media of communication, working together according to the user requirements. It is used to manage, command, direct or regulate the behaviour of devices or systems. Combining networks and control systems together facilitates the maintenance of the systems.

The result of this combination is referred to as the networked control system (NCS) [13] [14]. NCS is one of the main focuses in the research and industrial applications. Networked control systems are entirely distributed and networked control system used to provide data transmission between devices and to provide resource sharing and coordination management. These benefits inspired many industrial companies to apply networking technologies to manufacturing systems applications.

B. Communication Systems Architecture

Communication systems are designed to send messages or information from a source to one or more destinations. In general, a communication system can be represented by the functional block diagram shown in Fig. 2. The original telecommunication system was developed for voice communications.

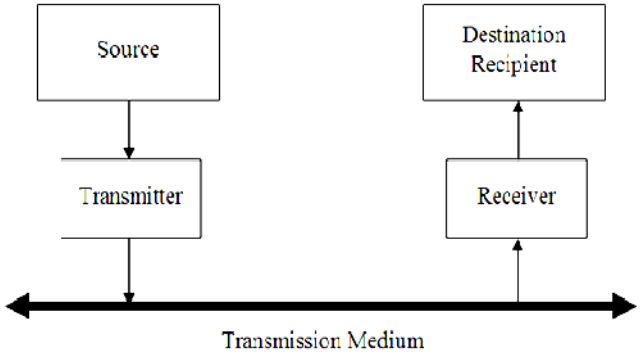

Figure 2. Functional Diagram of Communication System

Today communication networks include all types of voice, video and data communication over copper wire, optical fibers or wireless medium. Networks [15] [16] are organized into a hierarchy of layers where each layer has a well defined function and operates under specific protocols. The number of layers can vary from one network reference model to another but the goal of a layered structure remains common to all models. OSI model [17] is structured in a series of 7 layers, while the TCP/IP model includes only four layers, Fig. 3.

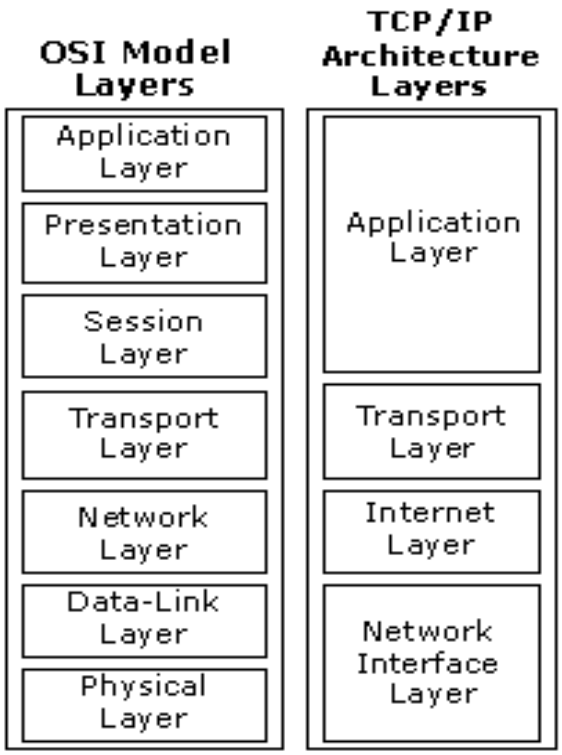

Figure 3. OSI and TCP/IP Reference Models

Each layer consists of hardware or software elements and provides a service to the layer immediately above it. With the advent of Internet, an increasing number of computer networks are now connected.

III. MODELING FORMALISM

Modeling communication protocols and distributed systems is not new. Several formalisms, simulators and modeling tools exist. However, one of the main challenges in modeling nowadays is establishing a precise relationship within a wide range of available modeling formalisms and comparing their descriptive power and verification capabilities. Thus, we propose to *unify* the modeling of both, protocols and distributed systems, in one formalism. In this way, our work eliminates the need to transform one formalism to another one and so facilitates the modeling process. Since time is an important feature of communication protocols, it is necessary to choose a formalism that allows to properly model the temporal behavior of distributed systems.

A. Formal Modeling

The Formal modeling consists of introducing system requirements (cost, security, manufacturing facilities, maintenance, evaluation, reliability and availability) into a small fragment. This introduction of the system requirements must be inside the chosen mathematical framework for the modeling process. The main purpose of a formal modeling is to clarify largely inexplicit information. During the construction of a formal model, many ambiguities must be removed. This consists, in general, of taking a decision. A good model is initially a model that one can understand easily and which can be explained simply. The procedures of verification must be simple and convincing. Several formal tools exist. However, *unified modeling language* UML, timed automata TA and Petri nets are some of the most used modeling formalisms that take into account time factor.

UML [18] is a well-defined, powerful formalism. However, UML lacks one important feature to achieve the desired needs since it has not a formal semantic and hence it is not possible to directly verify *timing requirements* which are *necessary in communication systems*. Timed automata are mainly used to model temporal applications and are not a general purpose formalism.

On the contrary, Petri nets are widely used for modeling concurrent and distributed systems. Many extensions and tools are available, mainly for time, identification of tokens and stochastic issues which are very important issues in communication protocols and services. So, the modeling and integration of all system elements (services and protocols) will be easier (no need to make a transformation). Table 1 shows the most important criteria that we used to choose the modeling formalism.

TABLE I. MODELING FORMALISM

Formalism	**UML**	**TA**	**Petri nets**
Method	Semi-formal	formal	formal
Time modeling	Recently	Yes	Yes
Application	General purposes	Hard timing	General purposes

In computer science Petri Nets are used for modeling a great number of hardware and software systems, and various applications in computer networks. A special advantage of Petri Nets is their graphical notation which reduces the time to learn Petri nets. This formalism has different extensions and tools.

Communication protocols have some specific characteristics and requirements. Thus, the tool selection criteria depends on the following requirements:

- *Time*: Communication protocols are real-time demanding applications. Transmitting and receiving data, accessing the channel, backoff procedure and other needs depend on time. Time Petri nets allow this feature.
- *Headers and Data fields*: Data packets have many fields which may be modeled as tuples. This feature is supported in high-level Petri nets.

- *Probabilistic and Stochastic Properties*: Messages exchanged over the network may be lost or perturbed, bit rate error is a function of noise, etc. The representation of such features can be made by stochastic functions. Stochastic Petri nets are a powerful model to handle such characteristics.
- *Sent and Received Packets*: Messages exchanged over the network require packet identification. Colored Petri nets have been proposed to satisfy this need.

The previous requirements have led us to use the high level Petri nets that combine all the aspects and power of the other extensions in one modeling tool.

B. Modeling FMS with Petri nets

Different modeling formalisms are used to model FMS. Petri nets and Automata theory are of the most used techniques to describe DES. Peterson [19] has introduced Petri nets to model and analyze the manufacturing systems. Other works like [20] have also proposed to model and analyze controllable DES. Later, Lin et al. [21] has introduced a decentralized control model. Zamaï et al. [22] has presented a hierarchical and modular architecture for real-time control and monitoring of FMS. However, newest modeling approaches are diverting towards the distributed modeling for the manufacturing systems [23].

Petri nets have been proposed by C. A. Petri in 1962 in his PhD thesis "Communications with Automata" [24]. Petri nets are a mathematical and graphical tool used for modeling, formal analysis, and design of different systems like computer networks, process control plants, communication protocols, production systems, asynchronous, distributed, parallel, and stochastic systems; mainly discrete event systems.

As a graphical tool, Petri nets provide a powerful communication medium between the user and the designer. Instead of using ambiguous textual description, mathematical notation difficult to understand or complex requirements, Petri nets can be represented graphically. The graphical representation makes also Petri nets intuitively very appealing.

A Petri net graph contains two types of nodes: Places "p" and Transitions "t". Graphically, places are represented by circles, while transitions are represented by rectangles, Fig. 4. Places and transitions are directly connected by arcs from places to transitions and from transitions to places. A place P0 is considered as an input place of a transition *t* if there is an arc from P0 to *t*. A place P1 is considered as output place of a transition *t* if there is an arc from *t* to P1.

Places can contain tokens represented by dots. These tokens are the marking of places. The initial marking of places is represented in the initial marking vector m0. The graphical presentation of Petri nets shows the static properties of the systems, but they also have dynamic properties resulting from the marking of a Petri net.

As a mathematical tool, a Petri net model can be described by a set of linear algebraic equations, linear matrix algebra, or other mathematical models reflecting the behavior of the system. This allows performing a formal analysis of the model and a formal check of the properties related to the behavior of the system: deadlock, concurrent operations, repetitive activities, etc.

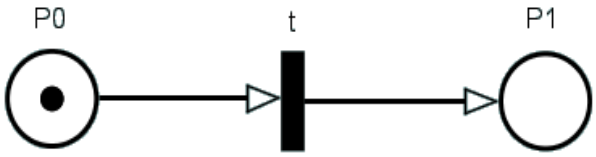

Figure 4. Discrete Event System

C. Proporties of our High-Level Petri Nets

In this subsection we will give a brief definition on the desired high-level Petri nets. This definition is not far from the definition of colored Petri nets [25]. However, we add to this definition a time notation.

Definition: A High-Level Petri Net is a tuple N= (P, T, A, m0, Σ, Λ, G, E, D) where:

- Σ is a finite set of non-empty color sets.
- Λ is a color function, Λ: P → Σ
- G is a guard function, G: T → Boolean expression, where:
  ∀t ∈ T: [Type (G(t)) = B_{expr} ∧ Type (Var (G(t))) ⊆ Σ], where:
  Type is the color type of the guard function,
  B_{expr} is a Boolean function
  Var is the variables of the guard function.
- E is an arc expression function, E: A→E(a), where:
  ∀a ∈A: [Type(E(a)) = Λ(p(a)) ∧ Type (Var (E(a))) ⊆ Σ], p(a) is the place of arc a.
- D is a delay function, D: E → TS, where TS is a delay associated to the arc inscription with the annotation symbol "@".

The arc expression function can contain any sign and/or mathematical or logical functions, such as programming language expressions. The delay function can be associated to both output arcs (from places to transitions) and input arcs (from transitions to places).

IV. COMPONENET-BASED MODELING

Component-based engineering [26] has a huge importance for rigorous system design methodologies. It is based on the statement which is common to all engineering disciplines: complex systems can be obtained by assembling components, ideally commercial-off-the-shelf (COTS) [27]. Reusability and extensibility are key factors that contribute to this success and importance. Component-based development aims at decreasing development time and costs by creating applications from reusable, easily connectible and exchangeable building blocks.

In component-based engineering research literature, several approaches [28] [29] have focused on the aspects of the development of components. However, reusing available, ready-to-use components decreases time-to-market for new systems and applications. This may be done by selecting the appropriate components from the available components

based on the needs and then assembling them to build a new component system-model.

Different methods of component specification software exist; from the Interface Description Language IDL (Object Management Groups' CORBA, java based components such as JavaBeans and Microsoft's .Net) to formal methods, by design-by-contract methods. Despite their widely difference in the details, they have a common concept: a component is a black box that is accessed through exposed interfaces. In this section we will state more precisely the main features that a component-based method must verify.

A. Genericity

Genericity is used in component-based engineering to reduce time-to-market, and raise productivity and quality in systems development. The term generic component refers to a component that implements a process or part of a process-set and that is adaptable to accommodate different needs. Genericity of a component is based on its independence compared to its type. This is an important concept for high-level methods because it can increase the level of abstraction of these methods.

A generic component can be seen as a parameterable element. Parameters should be specified and a specific version of the component (an instance) will be created and used. Another advantage is that a generic component can be represented as a *generic factory* that will create as many components as necessary for the application. Thus, the main objective of genericity is to integrate the component-based approaches with the technical approaches.

B. Modularity

Modular models are easier to design compared to similar complex models. "*Modularity is having a complex system composed from smaller subsystems that can be managed independently yet function together as a whole*" [30]. The objective of modularity is the ability to identify homogeneous, compatible, and independent entities to satisfy the needs of a system or an application. In many domains, modularity is essential to manage the design and the production of complex technology. Modular design aims at organizing complex systems as a set of components or modules. These components can be developed independently and then joined together.

The decomposition of a system model into smaller modules has the following advantages:

- A modular model can be very near to the real system, since it reflects the hierarchical structure inherent to the system.
- Components which are too complex can lose some of their details and their interactions can be confused. A component can be divided into smaller components until each module is of manageable size.
- It is possible to concentrate on each component as a small problem.
- Modular model allows testing each component separately.
- Implementation changes and corrections on simple components can be done easily.
- Documentation in modular structure becomes also easier.

C. Reusability

The implication of reusability is that the available components must give enough information to ease the assembly of components into a new system [31]. The information must include dependency and configuration information. To take sound decisions about selecting and reusing components, the following information is required:

- Operational specification: the semantic interaction of the component,
- Operation context: where and how the component will be used,
- Non-functional properties: describe the properties such as performance, security and reliability,
- Required interfaces and resources: the functionality and resources needed by the specified component to execute its main functionality.

Since all real systems are made of components, component-based systems are built of multiple components [32] that:

- are ready "off-the-shelf," (generally called "Commercial Off The Shelf"),
- have significant combined functionality and complexity,
- are self-contained and can be executed independently,
- will be used "as is" without modification,
- must be combined with other components to get the desired functionality.

All these benefits and more led us to use the component-based approach to model the distribution of manufacturing systems and the underlying protocols. The reuse of components is very important in the modeling level since most of the system parts and machines are the same. In addition, protocols share many properties. With the reuse of already modeled components, the time and modeling-cost are reduced. As we can see in Fig. 5, models are sharing some properties (the triangle). Once this part is modeled, it can be reused in any model that has a need to such component.

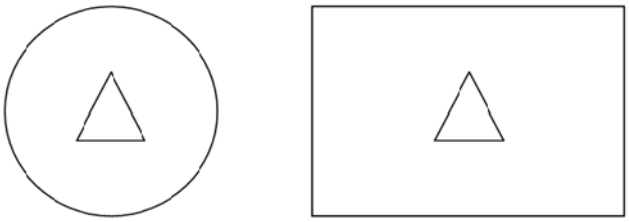

Figure 5. Basic Module Reusability

D. Components Abstraction

The modeled components are seen as black box where the internal functionality is hidden, while the interfaces

represent the service that can be offered by this component. Every component or module is characterized by its internal hidden behaviour. Its interfaces are chosen to reveal as little as possible about its inner implementation.

Components abstraction is useful for reducing the design complexity by decomposing a problem into connected components. Abstraction (or specification) describes the functional behavior of the components, i.e. components are considered to be specific to an application. Abstraction focuses on the important characteristics of component upon the designer point of view. This definition supports the abstraction of data, hiding internal function, reusability and self-contained component behaviour descriptions.

Thus, during the design of components we must focus on well-defining the service offered by the component at its interfaces and the parameters that can be adapted to the application requirements, rather than spending the time on describing its internal behaviour. This can be achieved by giving appropriate names to the interfaces and parameters and documenting these interfaces and parameters.

E. Components Interfaces

Components can be built according to the needs of the user and different requirements and points of view. However, these components are characterized by:

- The service they offer: each component has its own functionality and service. The resulting of this service depends on the parameters and value given to the component.
- The hidden implementation: the service and functionality are hidden. However, the designer has the access to the internal code but there is no need to modify the code.
- The interfaces: to access the component service or to connect the components, interfaces are used. Several modes of connection between the different components in the model can be defined.

The component interfaces declare the services that a component offers. They are used as an access point to the component functionality by other components. Since we use Petri nets to model the different component behaviors, we used *places* to be the input interfaces of components and the output interfaces are *transitions*. The input interfaces (places) receive as many tokens as the producer components. The output interfaces (transitions) generate as many tokens as the consuming components, Fig. 6.

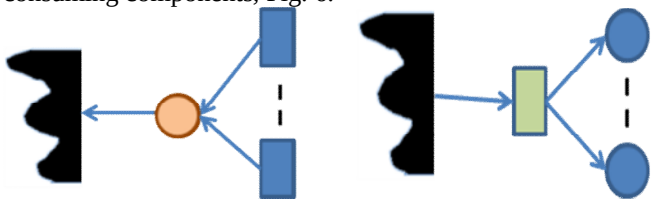

Figure 6. (a) input interfaces (b) output interfaces

This choice is coherent with the traditional way to model asynchronous communication between processes modeled by Petri Nets. Moreover it guarantees the genericity of the components and facilitates the connection between the different components.

The connection between interfaces of two blocks can be 1-to-many, many-to-1 or 1-to-1. As an example, Fig. 7 shows a many-to-1 and a 1-to-many connections. To illustrate the interest of this choice of interfaces, let us consider the modeling of workstations connected to a communications bus. A many-to-1 connection 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.

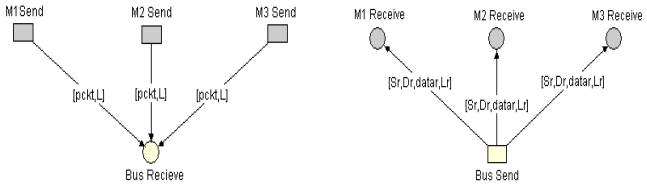

Figure 7. (a) input interfaces (b) output interfaces

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 workstations 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. . So this does not require any modification of the bus or the workstation component. Conversely, if the transitions are used as input interfaces and places as output interfaces, the addition of a new workstation would need to add a new token in the output place, and hence modify the internal code, so we loss the genericity.

V. MODELING COMMUNICATION PROTOCOLS

In our approach, we want to model reusable components. In this section, we will build the components that will be used to model the communication protocols. The modeling will be hierarchical since we build first the basic components. Then, with these components, we construct composite-components.

Before starting the construction of modeling components, we will analyze the data link layer protocols that we are interested in this work. These analyses will help to identify the basic common behaviors of the different protocols that lead to definition of the basic components. These basic components are the initial bricks of the library that will serve to model all the complete behavior of the different protocols.

A. A top-down analysis methodology

To build the basic components one must identify these components to be reused in different models. Since we are interested in manufacturing systems, the analyses will be made at the Data Link Layer protocols. The *Data Link Layer DLL* is the second layer in the OSI model. The data link

layer is often split in two sub-layers: the *logical link control LLC* and the M*edia Access Control MAC*, Fig. 8.

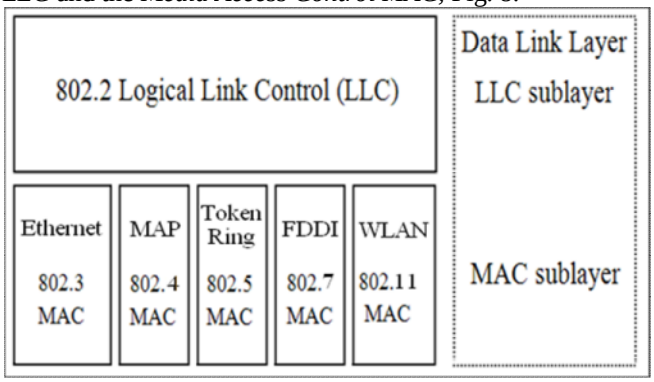

Figure 8. IEEE MAC Sublayer

The MAC sub-layer provides hardware addressing and channel access control mechanisms that enable hosts to communicate. Different MAC protocols are used. The data link layer defines most LAN and wireless LAN technologies: *IEEE 802.2 LLC, IEEE 802.3 Ethernet, IEEE 802.5 Token Ring, FDDI and CDDI, IEEE 802.11 WLAN.*

The next step is to define the protocols that have the same functionality. Here, one can find two protocols Ethernet IEEE 802.3 [33] and wireless IEEE 802.11 *Distributed Coordination Function DCF* [34] protocols that share the carrier sense multiple access CSMA procedure [35] to send the data over the shared medium. Finally, one must find the common behaviors to associate basic components to it. The result of these analyses is three basic common elements:

1) Channel check:

A workstation attempting to send data must at first check if the channel is free or not. Ethernet uses the CSMA/*CD* Protocol. Here, CD means collision detection. The workstation must check if the channel is free for a period of 9.6µs first, then it starts its transmission.

The IEEE 802.11 DCF uses the CSMA/CA protocol. Here CA means collision avoidance. To use the network, a workstation must before check if the channel is free for more than a period of time called Distributed Inter-Frame Space DIFS, Fig. 9. If so, the workstation starts a random backoff before starting its transmission. If the channel status is changed in both Ethernet and IEEE 802.11 deferring and backoff times, the workstation must restart the process of sensing the channel.

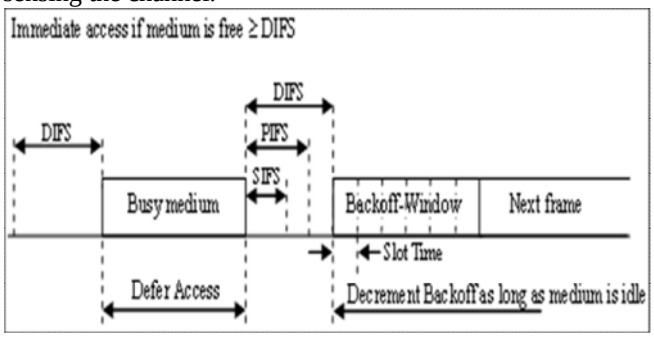

Figure 9. Channel Access in IEEE 802.11 DCF

2) Sending and Receiving: Data, Acknowledgments and JAM:

Workstations send and receive packets. These packets can be data packets, acknowledgment packets or JAM frame (a 32-bit frame, put in place of the correct MAC CRC). In Ethernet networks, workstations receive either a data packet or a JAM after a collision. The destination workstation does not need to send an acknowledgment to the transmitter at the MAC layer. However, in wireless LANs, the destination workstation must send an acknowledgment to the transmitter after a successful reception of a packet, Fig. 10. Otherwise, the transmitter will consider that its packet is lost or a collision has occurred, so it will retransmit this packet causing an extra load on network worthlessly.

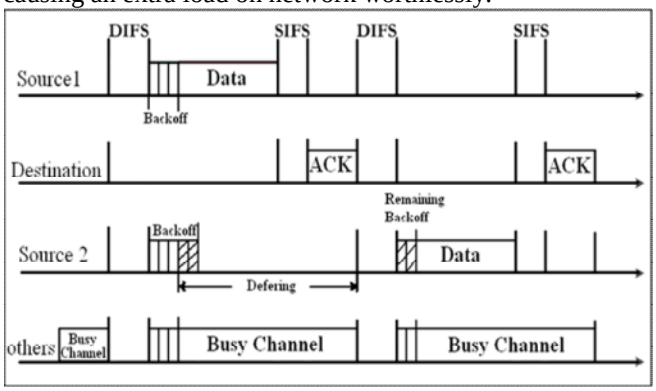

Figure 10. Backoff mechanism in IEEE 802.11 DCF without RTS/CTS

On the other hand, to send data, workstations need only to put the destination address in the packet. Since the medium is shared in most LAN technologies, all the workstations will see the packet. However, only the workstation that has the destination address reads the packet and the others will either forward it, or drop it.

3) Random and Binary Exponential Backoffs

In communication networks errors can occur. This is due to many factors like the surrounding environment, noise and interference, or because of collisions. Ethernet and IEEE 802.11 networks use the channel check and the inter-frame space to decide the medium access. Thus, collisions may occur when more than one workstation transmit on the shared medium at the same time. In Ethernet, the maximum time needed to send the first bit from one end to the other end of a 10BaseT medium is 25.6 µs. During this time, (an)other workstation(s) may attempt to send its data, as that the channel is considered as free.

As a result, a JAM signal is propagated over the shared medium informing the occurrence of a collision. Each workstation concerned by a collision starts a binary expositional backoff procedure, called *BEB*, to decide when it can do a new attempt to access the medium. The BEB algorithm computes randomly a waiting delay that increases with the number of the attempts *Tn* of the workstation.

At the beginning *Tn* equals zero (See Fig. 11). Each time a collision occurs, the workstation increments *Tn* counter until it reaches 15. Before trying to transmit its data again,

the workstation starts a BEB by taking a random value between 0 and 2^X and multiplies it by 51.2 µs, where:

$$X = \begin{cases} Tn, & \text{if } 0 < Tn \leq 10 \\ 10, & \text{if } 10 < Tn \leq 15 \end{cases} \quad (1)$$

This helps in decreasing the possibility for a collision occurrence. In case of no collision, the workstation continues transmitting and when it is done it leaves the channel. However, If *Tn* reaches 15, (the load on the channel is very high), then the workstation aborts its transmission and tries it again later.

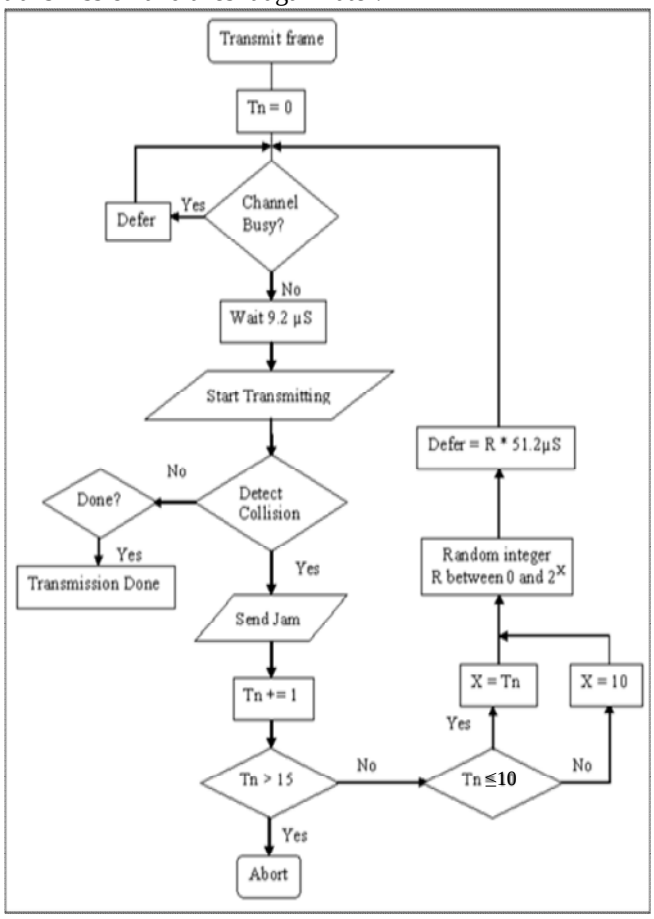

Figure 11. Transmission in Ethernet

In wireless LANs, after a collision, no JAM signal is sent. However, if the workstation does not receive an acknowledgment after a period of time equals to Short IFS *SIFS* (Fig. 9), it considers that a collision has occurred and starts a backoff procedure. For each retransmission attempt, the backoff grows exponentially according to the following equation:

$$ST_{backoff} = R(0,CW) * Slot\text{-}time \quad (2)$$

Where:

- *ST* is the backoff time.
- *CW* is the *Contention Window.*
- *R* is a random function.

In general, the initial value of CW (CW_{min}) is 16. After each unsuccessful transmission attempt, CW is doubled until a predefined maximum CW_{max} is reached (often 1024).

There are two major differences between Ethernet and IEEE 802.11 backoff processes:

1- The wireless LAN starts a backoff procedure even at the first attempt to send its data (Fig. 10), while Ethernet does not. This is one of the mechanisms used to implement the Collision Avoidance feature of CSMA/CA.

2- Ethernet starts its BEB algorithm after a collision (without conceding the status of the channel) and then restarts checking the channel to send its data. While in IEEE 802.11, the workstation checks first the channel status and then it decrements its backoff by:

$$R = \begin{cases} R - 1, & \text{if the channel is free during 1 time slot} \\ R, & \text{if the channel becoms busy} \end{cases} \quad (3)$$

The design of CSMA protocol offers fair access in a shared medium. This means that all the workstations have a chance to use the network and workstations cannot capture the channel for ever. The remaining value of R is reused after the channel status becomes free for more than a DIFS period. The workstation starts sending its data when R equals zero, Fig. 12.

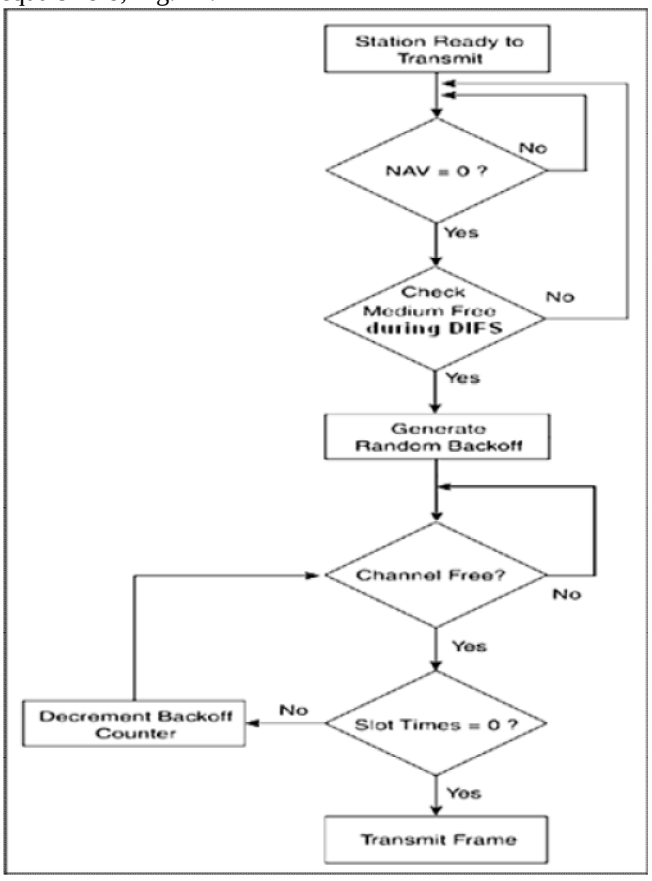

Figure 12. Medium Access Process for 802.11 Protocol

B. A bottom-up construction methodology

As one can see in the last subsection, three elements are in common. These elements can now be used to model the basic components.

1) The channel-check component

Fig. 13 shows a channel check component. Elements in light gray represent the places and transitions used to build the component. Elements in dark gray represent the interfaces of the component. Initially, the channel is idle for all workstations. This is represented by a token in place "Idle". A workstation that wants to send data (a token in place "Data send") must first check the channel.

In wireless LANs, the channel must be free for a period more than DIFS, while in Ethernet, it is 9.6 μs. This is represented by the '@t' at the arc between place "Idle" and transition "TF" (t' equals 9.6 μs in Ethernet and 50 μs in 802.11b). The workstation must wait before it starts transmitting, represented by a token put in place "sdata". In Ethernet the wait "@t" equals to 9.6 μs, while in 802.11 it is equal to random value between CW_{min} and CW_{max} slots time. Place "Backoff/Deferring Time" and transition "FC" is used to decrement the backoff in wireless LAN, while for Ethernet, it can be left as it is in the figure (no dependence to that transition in the model).

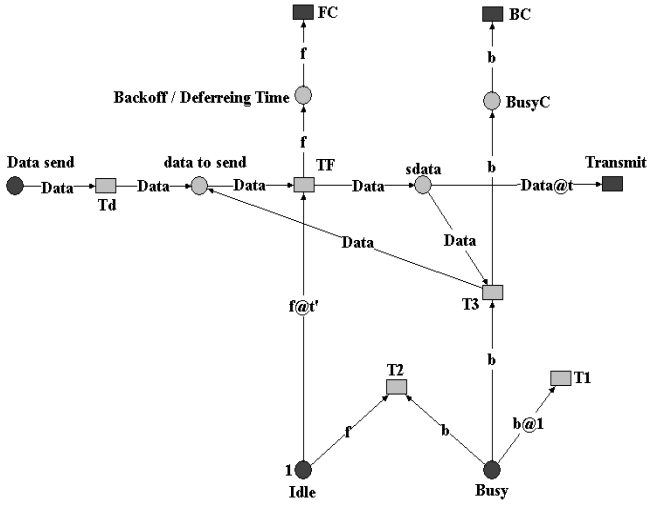

Figure 13. Transmission in Ethernet

Consequently, if the channel status is changed (a token is put in place "Busy"), the workstation can be in one of the following states:

- It is the transmitter (there is no more tokens in place "sdata"), then nothing is changed and the token in place "Busy" is consumed by transition T1;
- It attempt to send or it has no data to send, then T2 is fired;
- It is in the backoff/deferring phase, then T3 is fired (the workstation rechecks the channel again) and a token is put in place "BusyC" to stop decrementing the backoff. Hence, in wireless LAN, the workstation stops decrementing the backoff, but it keeps its remaining value.

In the three cases the channel status is changed from idle to busy.

Initially, this component has one token with value 1 (representing the free channel) in place Idle. The use of this component is possible in any protocol that demands the sensing the channel before transmitting data. It represents also the status of the channel free or idle. Let us notice here that, for genericity, we use two parameters t' and t to define the delay on the arc Idle-FT and arc Backoff/Deferring Time-Transmit.

2) Receiving and sending ACK component

Workstations receive two types of packets: data packet and ACK/JAM frames. In Ethernet network, no acknowledgment is sent after the reception of packet. Therefore, the received packet can be either a data packet or a Jam frame. While in wireless LAN, the received packet is either a data packet or an acknowledgment frame.

Fig. 14 shows the receiving and sending acknowledgment component. One assumes that a token is put in place "Receive". The fields of the token represents: the source address "Sr", the destination address "Dr", the received data "rdara" and the last field represents the lengths of the packet. The workstation checks at first the destination address "Dr" of the packet. The guard condition on transition "Address" checks if the received packet belongs to this workstation, a token is put in place "Data?". Otherwise, the token in place "Receive" is eliminated by transition "Drop". Hence, for simplicity, "Dr==1" is considered as the own address of the workstation, while "Dr==0" is used to represent the multicast or JAM frame reception.

Next, the guard condition of transition "ACK/JAM" is used to check if the received frame is an ACK frame or a JAM frame (for Ethernet only). The "abc" in the guard can be modified according to the needs of the designer and the type of network. However, if the received packet is a data packet, transition "DA" is enabled. This transition is fired after a time equals to the time needed to receive the packet modeled by the "@time(Lr)" at the outgoing arc. This "@time(Lr)" is a function that returns the time corresponding to the length "Lr" of the packet.

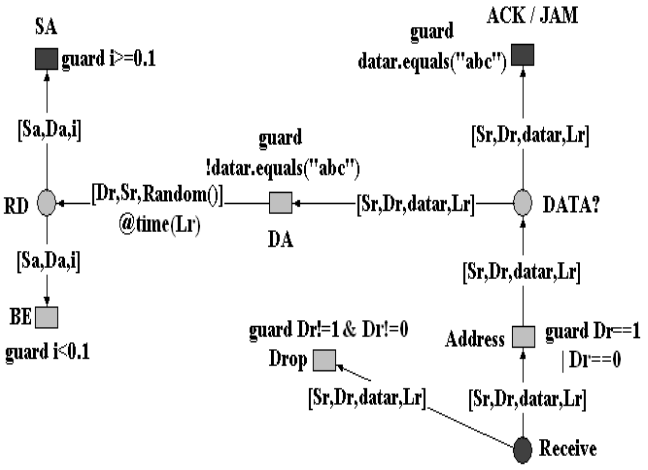

Figure 14. Receiving and Sending ACK Component

Let us notice here, the functions dynamicity can be used to model mobility of a wireless networks nodes. This can be done since the bit rate is a function of the signal strength and that the signal strength is a function of distance. This means if the source knows the location of the destination, then the distance can be computed, and hence the time needed to send a packet is determined.

The last step is to represent the bit rate or receiving errors. The random function *Random()* is used to generate a random variable i. Assuming that the bit rate error is less than or equal to 10% of the transmitted/received packets. So, if the value of i is less than 0.1, then the packet is discarded (the token in place RD is consumed by transition "BE"). Else, the packet is received correctly and then an acknowledgment is sent, by firing transition "SA". This interface can be left unconnected in Ethernet. As we can see in Fig. 14, the modification of tuples can be done easily, just by modifying the arc inscriptions according to our needs.

As one can see, this component has an important functionality since it is used to identify the received data (own or not), the type of the received data (JAM, ACK, data frame) and the process of sending an acknowledgment after a successful reception. Thus the use of this component is possible for the protocols demanding the identification of data and the send/receive process.

3) Backoff / BEB component

The third component is the backoff / BEB component shown in Fig. 15. As we can see in the figure, retransmitting the packet depends on the value of n, (transitions T6 and T7). If the packet is correctly sent/received (a token is put in place "Done"), then n is reset to z (0 for Ethernet and 1 for wireless), for the next attempt to transmit, place N. However, the component inscriptions depend on the type of the network. As an example, Table II shows the differences between Ethernet and IEEE 802.11b networks.

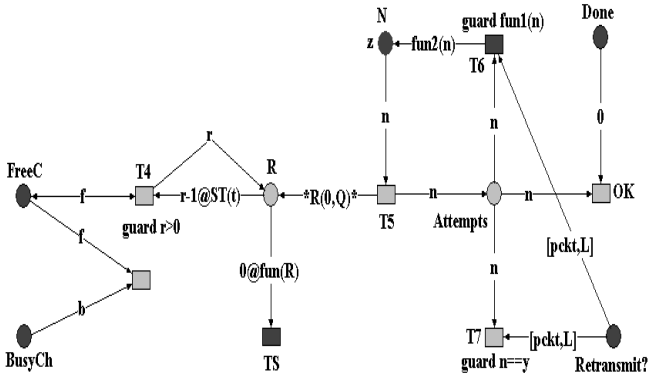

Figure 15. Backoff / BEB Component

In addition to Table II, in Ethernet, places "FreeC" and "BusyCh" are not used (they can be left as it is), since the backoff decrement in Ethernet does not depend on the status of the channel. While in 802.11b, this interface is very important in decrementing the backoff each time the channel is free for a slot time or the backoff is conserved if the channel status is changed to busy.

TABLE II. DIFFERENCES BETWEEN ETHERNET AND IEEE 802.11B NETWORKS

Variable Value	Ethernet	IEEE 802.11b
fun1(n)	n<15	n<33
fun2(n)	n=n+1	n=n*2
y	16	64
z	0	1
R(0, Q)	random(0, 2^X), X depends on n	random(0, CW)
Fun(R)	R*51.2µs	0
ST(t)	0	Time slot (20µs)

The firing of transition TS represents the (re)transmission allowance of a packet (backoff equals to 0). The backoff component is useful for the protocols that may need a specific timing procedure since it can be related to another components (which the case of wireless: by checking channel always) or just for standalone use

C. Application protocols

In this subsection, we will illustrate our modeling approach through two examples: IEEE 802.3 Ethernet MAC protocol and IEEE 802.11 MAC protocol because both protocols are based on CSMA. One of the objectives is to illustrate the advantage of having generic components and the hierarchical composition that allows building composite-components.

1) Modeling an Ethernet workstation

Ethernet is the most widely used LAN technology in the world. Ethernet was designed at its beginning at the Xerox Palo Alto Research Center PARC, in 1973. The used protocol differs from the classical protocols like token control, where a station cannot send before it receives an authorization signal, the token. With Ethernet, before transmitting, a workstation must check the channel to ensure that there is no communication in progress, which is known as the CSMA/CD Protocol.

Fig. 16 shows the detailed and complete module for the Ethernet workstation. As one can see in the figure, the three components: Backoff component, Channel Check component and Receive/Send component are reused to build the workstation. To complete the model and to bind the used components together, some additional places and transitions (in white) are used to answer the specification of an Ethernet workstation.

In the figure, one can see that five interfaces were not connected:

- The "FreeC" and "BusyCh" interfaces of the Backoff component, and the FC and BC interfaces of the Channel Check component, since Ethernet workstations decrement their backoff without the need to check whether the channel is idle.
- The SA interface of the Receive/Send component, because in this part we do not model the service offered by an Ethernet workstation

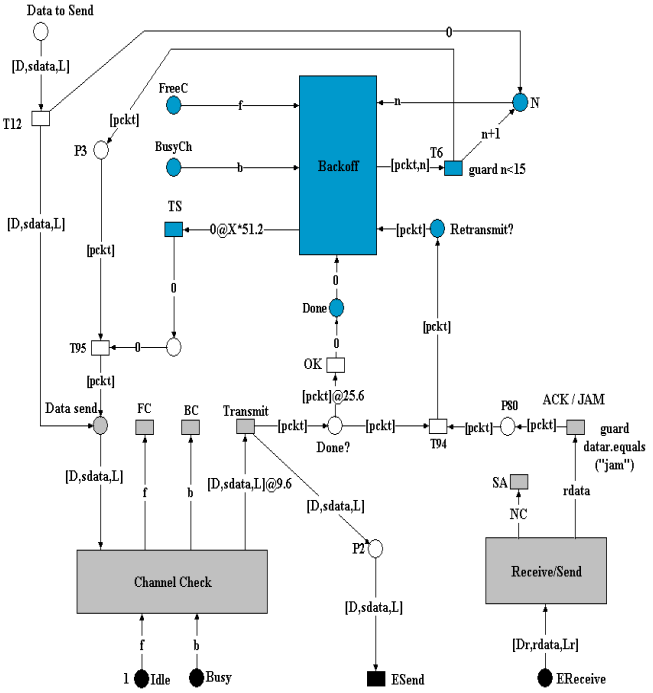

Figure 16. Composite design of an Ethernet Workstation Component based on Generic Basic Components

An important notice is that the whole component can be reused as one component for the Ethernet workstation to build a complete Ethernet network. In other words, this new component is seen as a composite-component with the black places and transitions as the interfaces of this new component.

2) Modeling a 802.11b DCF workstation

Wireless technology has become popular to access to 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. The IEEE 802.11 protocol is a member of the IEEE 802 family, which is a series of specifications for local area network technologies, (Fig. 8). IEEE 802.11 is a wireless MAC protocol for Wireless Local Area Network WLAN, initially presented in 1997. The IEEE 802.11 standard defines Medium Access Protocol and Physical (PHY) frameworks (layer 2 in the OSI model) to provide wireless connectivity in WLAN. This independence between the MAC and PHY has enabled the addition of the higher data rate 802.11b, 802.11a, and 802.11g PHYs. The physical layer for each 802.11 type is the main differentiator between them. However, the MAC layer for each of the 802.11 PHYs is the same.

Many other 802.11 variants have appeared. For instance, in 2004, the 802.11e was an attempt enhancement of the 802.11 MAC to increase the quality of service. The 802.11i and 802.11x were defined to enhance the security and authentication mechanisms of the 802.11 standard. Many other variants exist like 802.11c, 802.11d, 802.11h. The IEEE 802.11 MAC layer defines two coordination functions to access the wireless medium: A distributed coordination function DCF and a centralized coordination function PCF (Point Coordination Function).

Fig. 17 shows the detailed and complete module for the DCF IEEE 802.11b workstation model by the reuse of ready-to-use components designed from the previous sections. The workstation sets the value of N to 1 (place "N"), sense the channel (transition "TF"), sends its data (place and transition "Send") and waits for an acknowledgment (place "Wait"). If no acknowledgment is received during the SIFS period or 10µs, Transition T11 will fire putting a token in place "Retransmit?" to check if the packet can be retransmitted (transition T6) or not (transition T7).

As one can see in this figure, all the components are reused to compose the workstation module. All the interfaces were also used in this module.

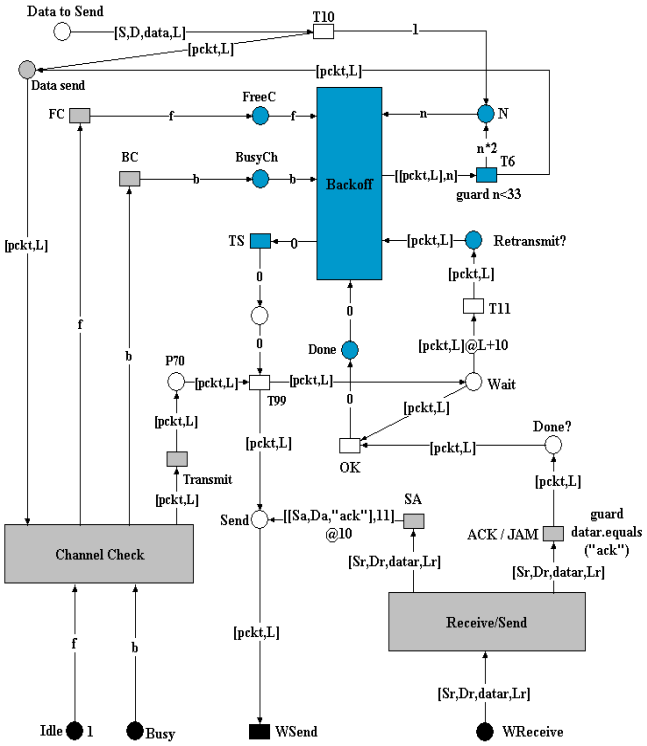

Figure 17. Hierarchical Design of a DCF IEEE 802.11b Workstation Component based on Generic Basic Components

VI. EXPERIMENTAL VALIDATION

In the previous sections, we have modeled several components (basic and composite components). In this section, we will validate and evaluate the quality and accuracy of our model by means of simulation. The obtained results will be compared with the data given by other studies about IEEE 802.11b network and also the results of NS-2 simulations performed in the same conditions.

A. Performance evaluation techniques

Different models and methods are used to evaluate the performance in communication and distributed systems [9]:

1) Measurement can offer the most exact and accurate results. The system is observed directly. However, measurement is often the most costly of the techniques since it is only possible if the system already exists. In some cases, measurements may not be accurate since it depends on the state of the system. For example, if network measurements are done during peak period, the observed results would not be the same as if the measurements were done during a period of low use of the network.

2) Analytical models may provide exact answers. Analytical modeling uses simple mathematical expressions to obtain the performance results for a system. However, these results may not be accurate, because of their dependencies on the made assumptions in order to create the model. The behavior of computer systems including processing periods, communication delays, and interactions over the communication channels is difficult to model. Analytical models are excellent to model small to medium systems that fulfil some requirements but this is not the case for industrial-sized, networked and distributed systems.

3) Simulation models [36] allow modeling and observing the system behavior. It facilitates the understanding of the real system. Simulation allows the evaluation of a system model in an executable form, and the data of such process are used to calculate different measures of interest. Simulation is generally used because the complexity of most systems requires the use of simple mathematical ways for practical performance studies. This makes simulation as a tool for evaluation. Simulation models allow creating very detailed, potentially accurate models. However, developing the simulation model may consume a big amount of time, but once the model is built it takes a little time to get results.

Table III shows a qualitative comparison between the different methods used to evaluate the systems performance.

TABLE III. COMPARISON OF THE DIFFERENT PERFORMANCE EVALUATION TECHNIQUES

Criterion	Analytical	Simulation	Measurement
Stage	Any	Any	Post prototype
Time Required	Small	Medium	Varies
Tools	Analysts	Computer Languages	Instrumentation
Accuracy	Low	Moderate	Varies
Trade-off evaluation	Easy	Moderate	Difficult
Cost	Small	Medium	High
Scalability	Low	Medium	High
Flexibility	High	High	Low

This comparison is based on different criteria [37] [38]:

- *Stage*: Which performance evaluation technique should be used at any point in life cycle,
- Time required: The time consumed/required by a particular technique,
- *Tools*: Which analytic tools, simulators, measurement packages are used,
- *Accuracy*: It represents the degree to which the obtained results match the reality (evaluates the validity and reliability of the results obtained).
- *Scalability*: It represents the complexity degree to scale a particular technique
- *Trade-off evaluation*: It represents the ability of a technique to study the different system configurations.
- *Cost*: This cost must not be considerable in term of time and money needed to perform the study.
- *Flexibility*: The system-model under test should be easy to modify and extend. The used evaluation technique should provide the possibility to integrate these considerations easily in the developed model.

Simulation seems to be the mostly used technique used to evaluate the performance of the computer systems. It represents a useful means to predict the performances of a system and compare them under many conditions and configurations. One major advantage of this technique is that even if the system is already implemented, it offers flexibility difficult to reach with measurement techniques.

Our modeling formalism, Petri nets, combines both the analytical and simulation models which let the possibility to model system mathematically. However, communication networks and distributed systems are so complex that building and solving the equations' system are too difficult and needs tools capable to perform this process.

More suitable for our aims are the discrete-event simulations [9]. Discrete-event simulation is a powerful computing technique for understanding the behavior of systems. In discrete-event simulations, the state changes occur at discrete steps in time. Discrete event simulation is mainly used in real environments such as communication networks and protocols [39], manufacturing systems [40], material handling [41], etc. General purpose programming languages like C/C++ and Java and several simulators such as NS-2 [42] and OPNET [43] are based on the discrete event simulation.

B. Simulations and Results

To perform the simulations, many tools and extensions of Petri Nets exist such as PROD, Renew, ALPHA/Sim, CPN Tools, Artifex and other tools [44]. However, the development of most of these tools has been stopped for a long time, or they do not support our needs or they are commercial. Two main, free of charge tools were possible to cover the previous features "CPN Tools" [45] and "Renew 2.1.1" [46].

However, during simulation, "CPN Tools" has shown an important problem that does not apply to our timing needs. We have chosen "Renew" since it is a Java-based high-level Petri nets discrete-event simulator.

Our simulations are based on full-mesh dense networks with different numbers of workstations:

i- The simulations were performed for different number of workstations sharing the medium.
ii- For each case, the simulations were repeated 100 times to get average measures.
iii- Each simulation assumes that all nodes transmit at 11Mbps.
iv- All nodes attempt to send data as soon as possible.
v- Each node has 1000 packets (to get the average possible measures) with average packet length of 1150 bytes (packet length varied from 800 byte to 1500byte).
vi- All simulations were accomplished on Intel® Core™ 2 Duo Processor T2300, 2G of RAM.

1) Average bandwidth per node

The first result is the average bandwidth per workstation. Fig. 18 shows the throughput of 802.11b nodes sharing a bandwidth of 11Mbps. As illustrated by the figure, the bandwidth per node decreases logically with the increase of nodes number. When the number of nodes is small each workstation has more bandwidth from the shared effective bandwidth. However, when the number of the nodes on the network increases, the bandwidth is decreasing exponentially. This is due to the increased number of collisions on the network, and so more bandwidth will be lost.

The other factor is that CSMA gives fair timing to the machines to access the channel. Thus, workstations must wait longer time to have access to the channel. Another factor is after a collision, the workstations must double their contention window which means longer backoff time. So, more time is spent to decrement the backoff or less total bandwidth.

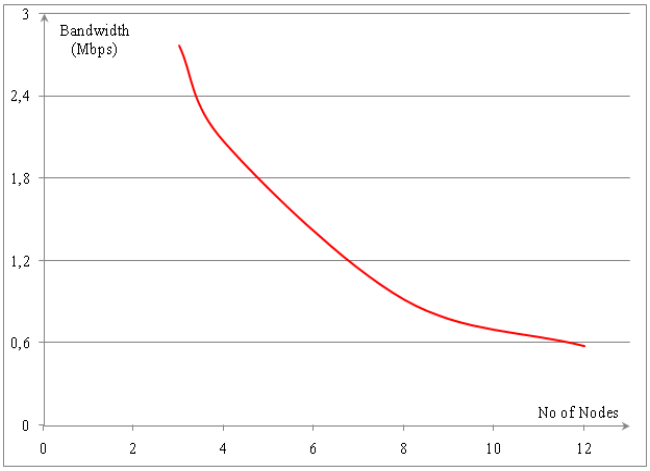

Figure 18. Bandwidth Variation with Number of Nodes

2) Collisions rate percentage

The next step is to compute the collision rate percentage or errors versus the network utilization. Fig. 19 shows how the collision rate increases when the number of workstations increases. As we can see in the figure, when three workstations are sharing the medium, the collision rate is nearly 8%. However, when there are 12 workstations sharing the medium, the collision rate reaches 23.2%. These results confirm the results obtained in the previous section and our explanation.

As one can see, the collision rate is increasing linearly until certain point (8 workstations). The reason is when more workstations attempt to send, more packets are on the shared channel and hence the probability that a collision occurs increases. However, when the number increases more, the collision rate increase becomes slower. The explanation for this *evolution* is the *backoff procedure*. With more workstations, the number of collisions increases, and the value of CW also increases (backoff time). On the other hand, this increment of backoff time decreases the probability of a collision, since workstations in collision must wait for longer time before attempting to send again. So, the collision rate increment becomes slower.

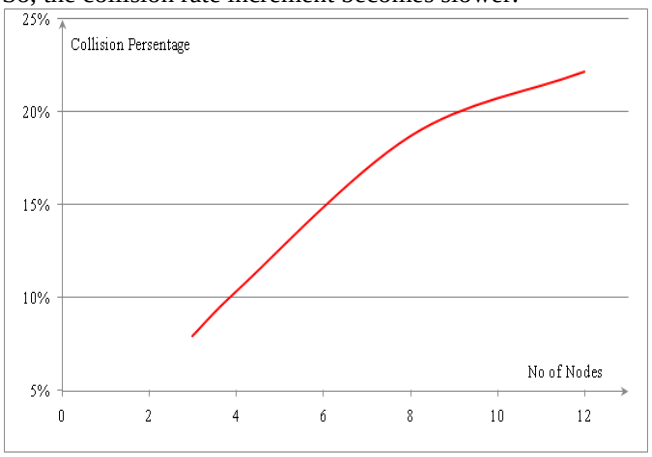

Figure 19. Collisions Rate Percentage

3) Transmission Time per Packet

The next test is to measure the overall time needed to send a packet over Ethernet or DCF protocols (from sender side to receiver side). Fig. 20 shows the time required to transmit one packet versus the number of nodes on the network. The transmission time increases linearly due to the increased number of sent packets on the network and collision rate.

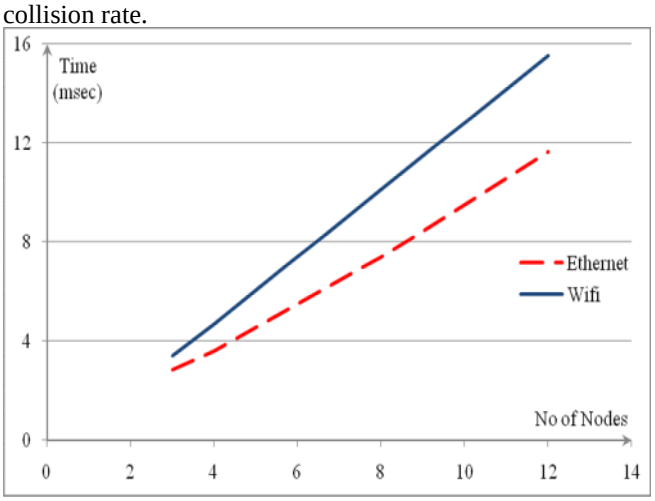

Figure 20. Transmission Time per Packet

However, sending a packet over Ethernet requires less time than sending it over DCF. The figure shows that with three nodes on the network, DCF seems to be the same as Ethernet. However, with the increase of nodes the difference becomes obvious. This is due to:

1- A workstation attempting to use the channel in wireless networks needs to ensure that the channel is idle during a DIFS period or 50µs, while in Ethernet it only needs 9.6 µs.
2- From the first attempt to transmit, wireless nodes starts a backoff procedure (Bavg = 8 * 20 µs) decremented only if the channel is idle, while in Ethernet, workstations defers only for 9.6 µs.
3- After a collision, in wireless networks, the channel status becomes idle only when all the workstations finish their transmissions (no collision detection process), while in Ethernet the channel becomes idle after 51.2 µs (channel acquisition slot time).
4- The backoff procedure used after each collision in wireless networks doubles the contention window value which is already 8 times greater than the one used in Ethernet. This makes the backoff in wireless greater than Ethernet BEB even with slot time (20µs) less than the 51.2 µs used in Ethernet.

C. Comparison with ns-2 simulator and other studies

To evaluate the quality and accuracy of our model, we have used the network simulator NS-2 as a comparative tool since it is widely used to model communication protocols. The NS-2 simulator is a discrete-event network simulator that allows simulating many scenarios defined by the user. It is commonly used in the research due to its extensibility, since it is an open source model. NS2 is widely used in the simulation of routing, multicast protocols and ad-hoc network.

Fig. 21 shows the results obtained from NS-2 and those from our model, (Fig. 18). As we can see the results of both simulations Renew and NS-2, are nearly identical which confirms the correctness of our model. Moreover, if we compare our obtained results with those in [Anastasi05] and [Heusse03], we can get also the same results from both the simulation technique and the equation we obtained from the results.

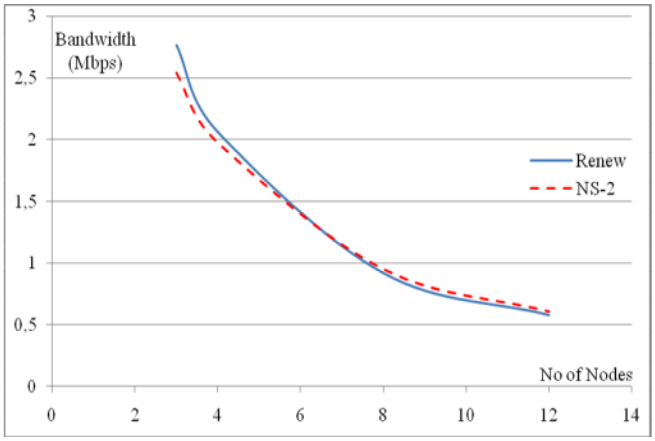

Figure 21. Comparison between our model and NS-2

The other comparison is the effective simulation time. As we can see in Fig. 22, the simulation time increases in a linear way when the number of nodes increases (confirmed by the results in Fig. 20). The figure shows that NS2 needs less time to perform the same simulation. However, NS2 does not support the step-by-step simulation to verify the system event by event. The second important issue is that it is not possible to model distributed services with NS2 (no supporting package). However, with "Renew" as Petri nets editor and simulator, it is possible to combine both services and protocols in one global model.

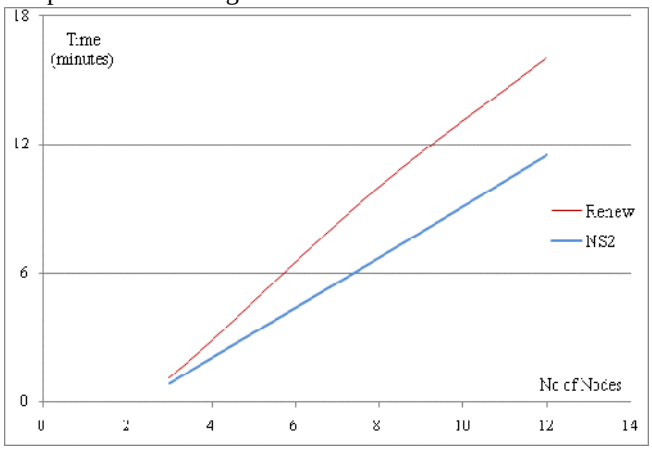

Figure 22. Effective Simulation Time versus number of nodes

VII. Case Study: Evaluating Performance of a Distributed Manufacturing System

In the last sections, we have shown the modeling part of the communication protocols. In this section we will show the modeling part that concerns the services. An illustrative example, Fig. 23, will be used to model the services offered by a production system. The used modeling technique will be the same as the communication protocols, i.e. component-based methodology, where each part of the system is modeled in hierarchical composition: "*service-workstation*", i.e. each service is modeled over a workstation.

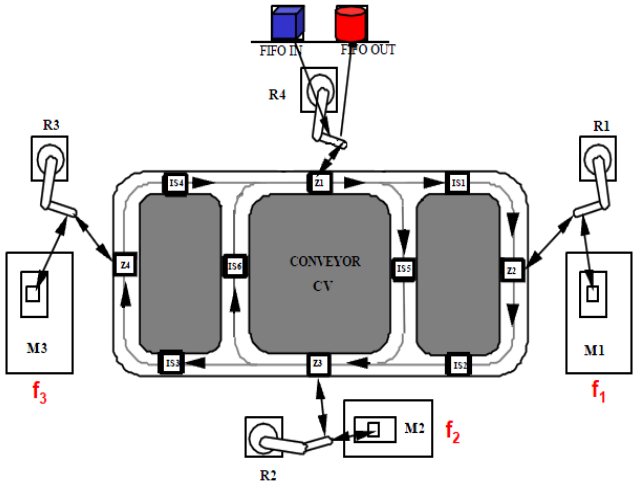

Figure 23. Manufacturing Plant with Flexibilities

A. System Components

Fig. 24 shows the complete system components used to transfer one product from *IN/OUT* area to any machine *M1*, *M2* or *M3*. *Z1* to *Z4* represent the input and output areas for each machine and the IN/OUT area. The capacity of each is limited to one product. *IS1* to *IS6* areas represent the stock area before and after machining a product. The capacity of each stock area is greater than one. R1 to R4 represent the robots used to make a transfer from a Z area to a machine or IN/OUT area and vice versa. Finally, *T1* to *T8* represent the transfer elements from and to a Z and an IS stock areas. The component represents the service offered by a resource, a robot or a transfer component. Machines, Z's and stock areas are considered as shared resources.

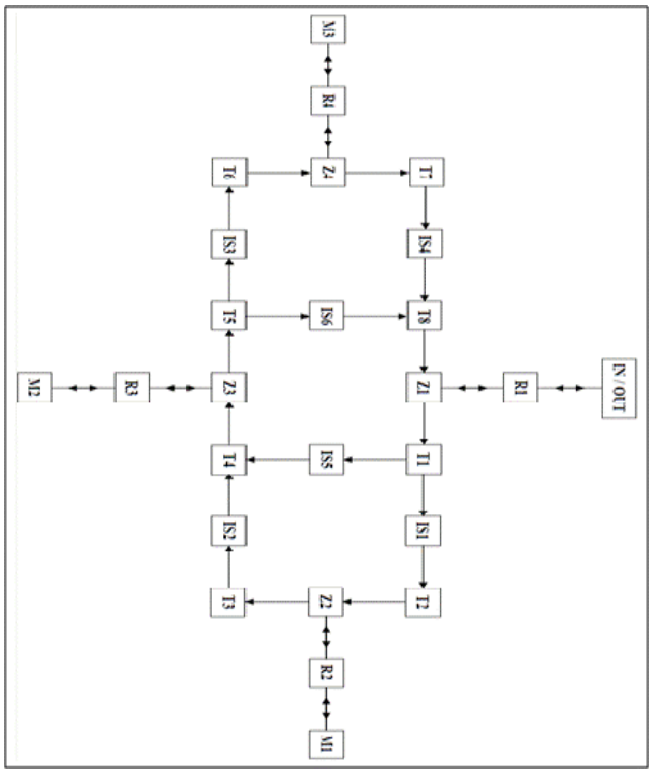

Figure 24. Complete Area and Transfer Components

B. Product transfer

In order to transfer a product from one area to another one, areas must allocate the required area/resource. Pre-allocation is passed through a transfer component. Transfer components check the possibility to allocate the destination area (depending on the capacity of each area). An acknowledgement is from the destination area when a place is free. During this time the source area and the transfer component are in waiting period (machines and Z areas do not perform any action during this time, while stock areas can receive products from other components).

Fig. 25 shows the centralized model of a product transfer from S to D areas. In the figure, to transfer a product, the product must be available in area S (a token put in place S/REQ), the transfer component must be also available (a token in place S→D/NOP) and a free place in area D (a token in place D/CONS). These three tokens enable the transition S→D/t1 and a token is then put in place S→D/TRSF-START starting the transfer process. The transfer component takes the product from area S. The firing of transition S→D/t2 and the put of a token in place S/ACK inform that a place is released up in area S. The transfer process continues by putting a token in place S→D/TRSF-END. When the product arrives to area D (transition S→D/t3), the transfer component becomes free again (a token is put in place S→D/NOP) and an area is used in area D (a token is put in place D/PROD).

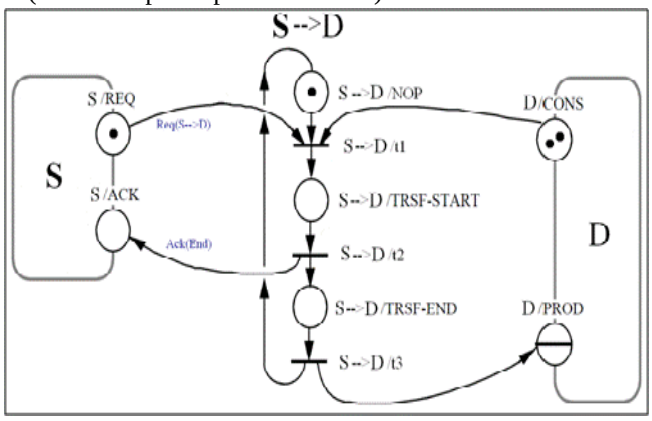

Figure 25. Product Transfer in centralized model

Fig. 26 shows the complete messages exchanged in case of implementation of the 3 processes (S, S→D, and D) on 3 different computers.

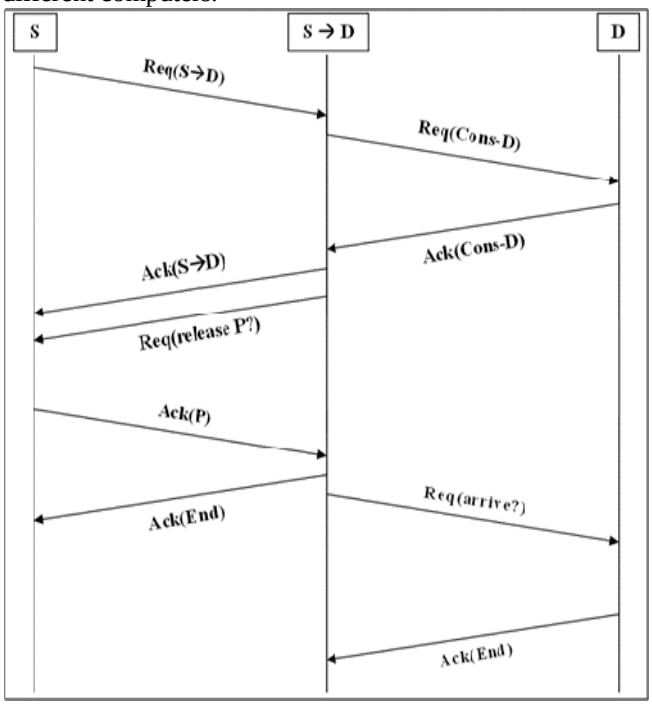

Figure 26. Exchanged Messages over the Network for the Transfer

Each process plays a different role with regard to the client/server mechanism. S is always a client and D is

always a server. The role of S→D varies depending on the message. At first, the source area *S* (workstation) sends a request message to the transfer workstation, *S→D* (T_i or R_i), containing the destination workstation *D*. T_i (or R_i) sends a request to D, requesting a free place (*Cons-D*). If there is a free place, *D* will send a positive acknowledgment to T_i (or R_i), otherwise S and T_i (or R_i) will stay in a waiting period. Once T_i (or R_i) receives the acknowledgment, it sends two messages to S containing a positive acknowledgment and a request to release the product. When the product is released S sends an acknowledgment to T_i (or R_i) to start the transfer. When T_i (or R_i) takes the product, it sends an end message to S to free one its places (Cons-S). Finally, it sends a message to D asking the arrival of the product to its side. Once the product arrives to D, it sends an acknowledgement to T_i (or R_i) informing the end of the transfer.

C. Modeling the service components

Fig. 27 shows the complete messages sent and received by a transfer component. The component receives a request packet from an area (transition t60). In order to validate this request a token must be present in place "Cons", representing the capacity of this component. It sends a message to the destination area requesting a free place. The component stays in a waiting period (place p60) until it receives acknowledgment packet from the destination area (transition t62).

Once the acknowledgment arrives, the transfer component sends request ("release P?") to the source area. To insure a proper functionality of this module, a guard is associated to the transition t63 to assure that the sender of the acknowledgment is the destination area. Again, the component stays another time in waiting period (place p62) until it receives the second acknowledgment.

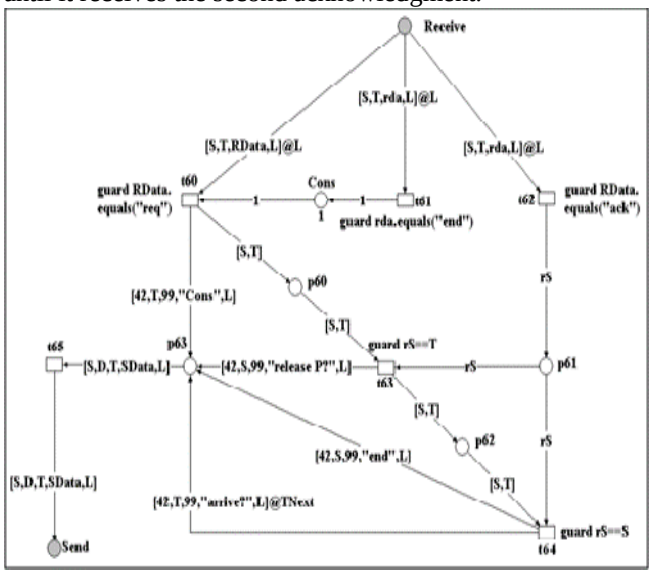

Figure 27. Transfer Component – Service Part

When the acknowledgment arrives, transition t64 is enabled (condition: the sender must be the source), two messages are sent: to the sender releasing one place (the product is taken by the transfer component), and to the destination area requiring if the product has arrived. This second message is sent when the first message is sent (the *TNext* inscription on the arc between t64 and p63).

Fig. 28 shows a complete Petri Net model for Z1 area. In the figures, when a product arrives to the area, a procedure of exchanged messages starts depending on the destination area until that product is transferred to its final destination. Hence, only the workstation component is connected to the network, while the output interfaces of the service component (dark gray) is connected directly to their destination components.

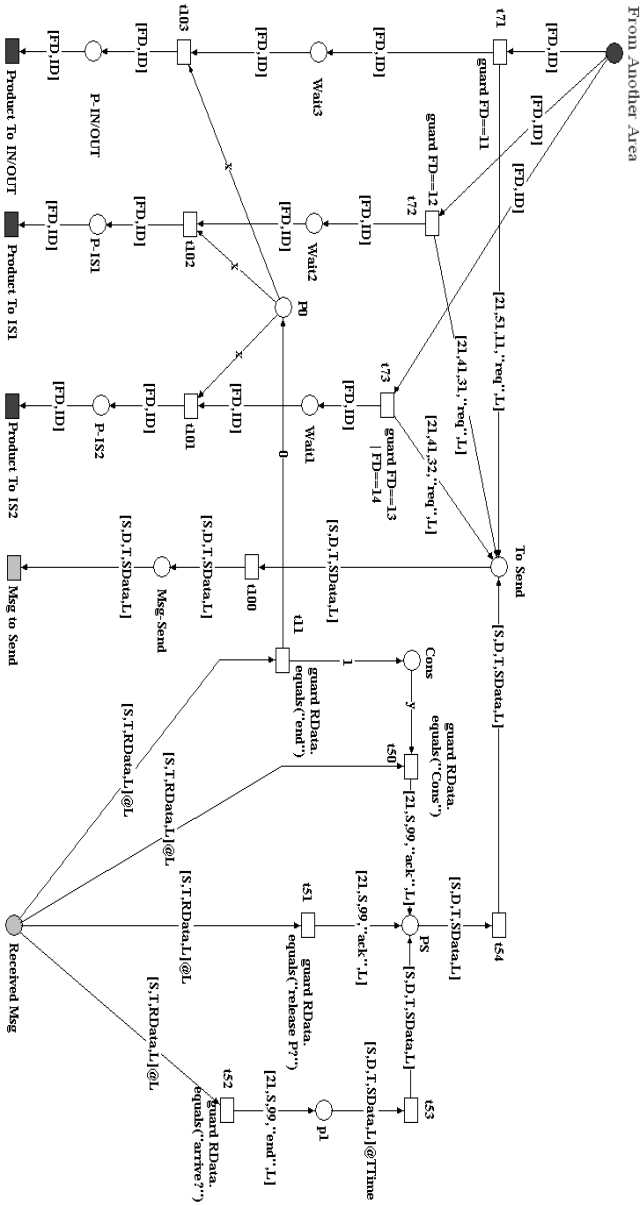

Figure 28. Hierarchical Service-Workstation Petri Net

Fig. 29 shows a sub-model for a transfer of a product from area Z1 to stock IS1 through the transfer element T1. In the figure, the modeled components are reused to build the

whole system. This reuse is applicable for the other elements in the system. Some additional places and transitions are used to complete the system-model and to insure its functionality.

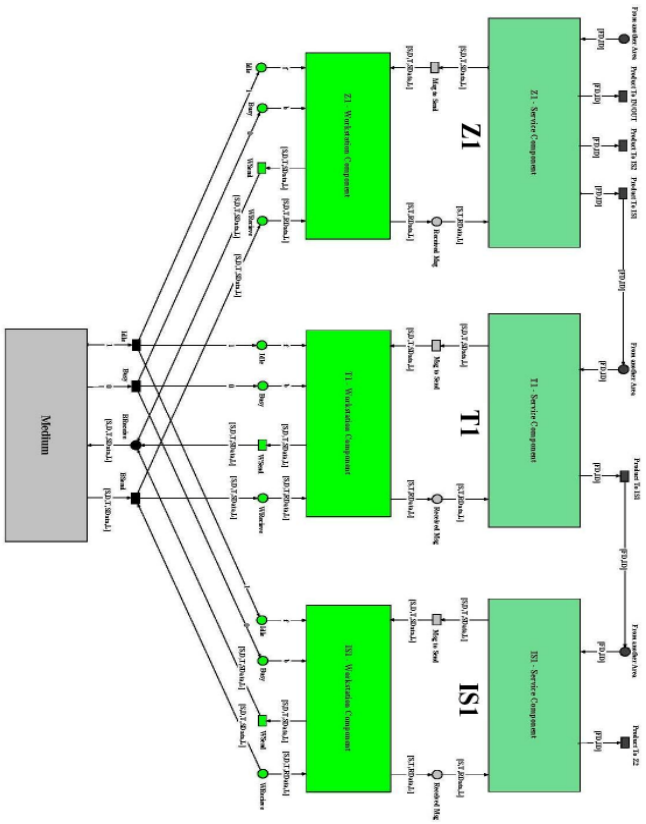

Figure 29. Service-Workstation Composite-Component

D. Simulating the complete model

The simulation was performed on the same PCs used in the above sections. The system is assumed to perform 100 different products. The simulation aims to see the impact of using different type of products and different protocols over the system. The transfer time is supposed to be 50 msec and the machining time to be 100 msec. These values have been chosen in milliseconds to really verify the impact of the underlying network on the system. Otherwise, if we use the real values in minutes, the impact of the underlying network would not be obvious with the example we have used. The number of simultaneous products per type is varied from 2 to 5 products.

The type of services on the system affects the number of exchanged messages and transactions on the network. For example, to perform the service f2, the number of transactions is 72 exchanged messages per product. However, to complete service f1 or f2, the number of exchanged messages is 90 messages per product. This is in the case of one product only on the system. However, when there are several products on the system, this number increases due to collisions. So, this number may reach 90~100 messages per product for service f2, and 110~120 messages per product for service f1 or f3.

a) One Product

The first simulation is performed to get an idea about the time needed to machine one product over the system. Table IV shows the impact of changing the communication protocol in the system over the time needed to finish one product. An important difference appears between Ethernet at 10Mbps and 100Mbps. However, the 1Gbps does not create a big difference, since the machining and transfer times are the dominant in this case.

TABLE IV. TIME TO MACHINE A PRODUCT

Service	802.11b	E-10Mbps	E-100Mbps	E-1Gbps
f2	564.5 ms	567.6 ms	506.7 ms	500.7 ms
f1 or f3	680.2 ms	684.5 ms	608.5 ms	600.9 ms

The other interesting result is the time difference when the required service is f2, or f1 or f3. Since the path to finish the product is longer, the time needed to make the product is clearly longer. In this part, 11Mbps 802.11b seems to be better than 10Mbps Ethernet.

b) Same Products; Different Protocols

The second results are the most important, since they show the impact of changing the communication protocol on the system. Different remarks can be done from the Fig. 30:

1- 802.11b protocol does not present a good choice. This result is conforming with the results of Fig. 20. This becomes clear when the number of simultaneous products increases (the number of exchanged messages increase also).

2- A big time difference is noticed when using 100Mbps Ethernet (compared to 10Mbps Ethernet and 802.11b). The number of messages is important. With 2 simultaneous products of each type, the number of exchanged messages reaches 500 to 600 exchanged messages. With 3 simultaneous products of each type, the number of exchanged messages reaches 900 to 1000 exchanged messages. While with 5 simultaneous products of each type, there are nearly 1400 to 1500 exchanged messages on the network.
The type and speed of protocols is very important since to exchange this huge number of messages on the network, the bit rate is very important and decreases obviously the time needed to exchange these messages between the different resource/workstation on the system.

3- The use of 1Gbps Ethernet did not show a big difference with respect to 100Mbps Ethernet. However, this conclusion is not really correct. The impact of using Giga Ethernet can appear if the modeled system is larger (more machines, stock areas, resources, etc.).
In that case, the number of exchanged messages over the network will be greater. Thus, the impact

of using Giga Ethernet will become obvious since the time needed to send these messages will be shorter (for example, as the time difference between 10 and 100Mbps).

However, in our model the number of modeled components is still medium (3 machines, 4 resource areas and 6 stock areas). So, the machining and transfer times are dominant here when using Giga Ethernet compared to 100Mbps Ethernet.

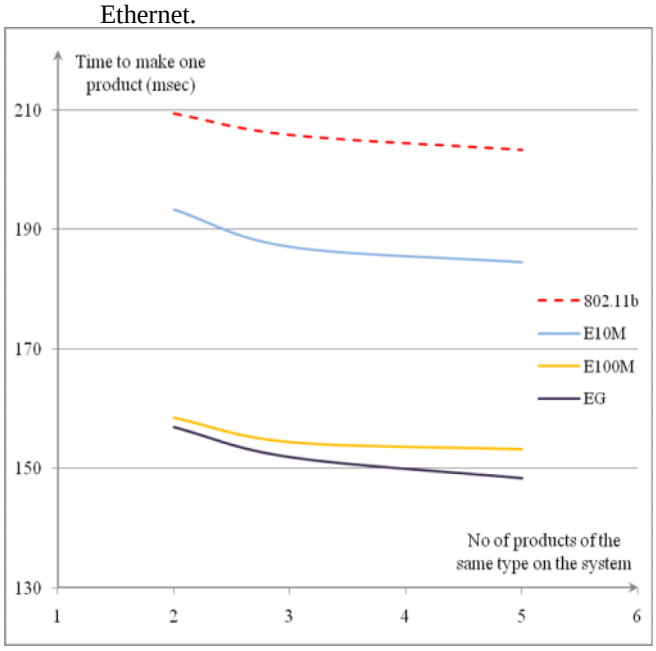

Figure 30. Impact of changing the communication protocol in the system

VIII. CONCLUSION

Distributed systems are more and more present in our daily life. These systems are complex and can be in one place or even everywhere in the world. The use of distributed systems allows sharing different and expensive resources by a several clients. Thus, the need to control the distributed systems is very important. Manufacturing systems are one kind of these systems.

The need to model these systems before their implementation is important. The design stage allows verifying some of their properties. A well-designed model that takes into accounts all the requirements and constraints of a system can save cost and time.

In this work, we have presented the problem of modeling manufacturing systems and the underlying communication protocols. However, modeling a huge and complex system implies to have also a big and complex model. So, we have proposed in this work a component-based modeling approach based on High-Level Petri Nets.

This approach can meet the challenges of modeling the distributed systems and the communication networks. Genericity, modularity and reusability are the main and important characteristics of this approach since it allows reusing ready-to-use components and easily fitting them to new system-models depending on the requirements of clients and applications. These advantages allow building complex models in an easier way.

REFERENCES

[1] A. Tanenbaum, "*Distributed Operating Systems*". Prentice Hall, 1995.

[2] G. Coulouris, J. Dollimore, and T. Kindberg, "*Distributed Systems: Concepts and Design*", 3rd ed. Pearson Education, 2001.

[3] A. Toguyeni, "*Design of Modular and Hierarchical Controllers for Reconfigurable Manufacturing Systems*." IMACS Multiconference on Computational Engineering in Systems Applications, Vol. 1, pp. 1004-1011, 2006.

[4] H. Sarjoughian, W. Wang, K. Kempf, and H. Mittelmann, "*Hybrid discrete event simulation with model predictive control for semiconductor supply-chain manufacturing*." Proceedings of the 37th Conference on Winter Simulation, pp. 256 – 266, 2005.

[5] P. Berruet, J. Lallican, A. Rossi, and J-L. Philippe, "*A component based* approach for the design of FMS control and supervision." IEEE International Conference on Systems, Man and Cybernetics, Vol. 4, pp. 3005-3011, 2005.

[6] A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "*Performance Analysis of IEEE 802.11b Wireless Networks with Object Oriented Petri Nets*", Electronic Notes in Theoretical Computer Science, Proceedings of First International Workshop on Formal Methods for Wireless Systems FMWS'08/CONCUR'08, Vol. 242, No 2, pp. 73-85, Canada, 2008.

[7] A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "*Network Protocol Modeling: A Time Petri Net Modular Approach*." 16th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2008, pp. 274-278, Croatia, 2008.

[8] A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "*A Component Modular Modeling Approach Based on Object Oriented Petri Nets for the Performance Analysis of Distributed Discrete Event Systems*," Fifth International Conference on Networking and Services ICNS, pp.222-227, Spain, 2009.

[9] C. Cassandras and S. Lafortune, "*Introduction to Discrete Event Systems*", 2nd ed. Springer Science+Business Media, LLC, 2008.

[10] P. Ramadge and W. Wonham, "*The Control of Discrete Event Systems*." Proceedings of the IEEE, Vol. 77, No. 1, 1989.

[11] P. Paraskevopoulos, "*Modern Control Engineering*". Marcel Dekker, Inc., 2002.

[12] R. Burns, "*Advanced Control Engineering*". Butterworth-Heinemann, 2001.

[13] W. Zhang, M. Branicky, and S. Phillips, "*Stability of Networked Control Systems*." IEEE Control Systems Magazine, Vol. 21, pp. 84-99, 2001.

[14] F. Wang and D. Liu, "*Networked Control Systems: Theory and Applications*". Springer-Verlag London Limited, 2008.

[15] N. Mir, "*Computer and Communication Networks*". Prentice Hall, Inc, 2007.

[16] W. Stallings, "*Data and Computer Communications*", 8th ed. Prentice-Hall, Inc, 2007.

[17] H. Zimmermann, "OS1 Reference Model-The IS0 Model of Architecture for Open Systems Interconnection." IEEE Transactions on Communications, Vol. COM-28, No. 4, 1980.

[18] Unified modeling language UML : *http://www.omg.org/uml* - 2009.

[19] J. Peterson, "*Petri Net Theory and the Modeling of Systems*". Prentice-hall International, 1981.

[20] P. Ramadge and W. Wonham, "*Supervisory control of a class of discrete event processes.*" SIAM Journal on Control and Optimization, Vol. 25, No. 1, pp. 206-230, 1987.

[21] F. Lin and W. Wonham, "*Decentralized supervisory control of discrete event systems with partial observation*." IEEE transaction in Automatic Control, Vol. 35, pp. 1330-1337, 1990.

[22] E. Zamaï, A. Chaillet-Subias and M. Combacau, "*An architecture for control and monitoring of discrete events systems*." Computers in Industry, Vol. 36, No. 1 - 2, pp. 95-100, 1998.

[23] J. Petin, P. Berruet, A. Toguyeni, and E. Zamaï, "*Impact of information and communication emerging technologies in automation engineering : outline of the intica proj*." 1st Workshop on Networked Control System and Fault Tolerant Control, 2005.

[24] C. A. Petri, "*Communication with Automata*." Technical Report RADC-TR-65-377 Rome Air Dev. Center, New York, 1966.

[25] K. Jensen, "*Coloured Petri nets: A high level language for system design and analysis*", Lecture Notes in Computer Science, Vol. 483/1991, pp. 342-416, 1991.

[26] P. Brereton and D. Budgen, "*Component-Based Systems: A Classification of Issues*." IEEE Computer, Vol. 33, No. 11, pp. 54-62, 2000.

[27] D. Carney and F. Long, "What Do You Mean by COTS? Finally, a Useful Answer." IEEE Software, 2000.

[28] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis, "*An Approach to Modelling and Verification of Component Based Systems*." Lecture Notes in Computer Science, SOFSEM 2007: Theory and Practice of Computer Science, Vol. 4362, pp. 295-308, 2007.

[29] R. Bastide and E. Barboni, "*Component-Based Behavioural Modelling with High-Level Petri Nets.*" In MOCA'04 Aahrus, Denmark, DAIMI, pp. 37-46, 2004.

[30] R. Langlois, "*Modularity in technology and organization*." Journal of Economic Behavior & Organization, Vol. 49, p. 19–37, 2002.

[31] C. Geisterfer and S. Ghosh, "*Software Component Specification: A Study in Perspective of Component Selection and Reuse*." Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, pp. 100-108, 2006.

[32] A. Brown and K. Wdlnau, "*Engineering of Component-Based Systems*." Second IEEE International Conference on Engineering of Complex Computer Systems, 1996.

[33] IEEE 802.3 Ethernet Working Group: *http://www.ieee802.org/3/* - 2009.

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

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

[36] P. Fortier and H. Michel, "Computer Systems Performance Evaluation and Prediction". Digital Press , 2003.

[37] A. Chhabra and G. Singh, "*Parametric Identification for Comparing Performance Evaluation Techniques in Parallel Systems*." Proceedings the 1st National Conference on Challenges and Opportunities in Information Technology, COIT, pp. 92-97, 2007.

[38] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling ". John Wiley & Sons,Inc., 1991.

[39] H. Lim, B. Wang, C. Fu, A. Phull, and D. Ma, "*A Middleware Services Simulation Platform forWireless Sensor Networks*." The 28th International Conference on Distributed Computing Systems Workshops, ICDCS, pp. 168-173, 2008.

[40] N. Kumar and R. Sridharan, "Simulation modelling and analysis of part and tool flow control decisions in a flexible manufacturing system ." Robotics and Computer-Integrated Manufacturing, 2009.

[41] V. Giordano, J. Zhang, D. Naso, M. Wong, F. Lewis, and A. Carbotti, "*Matrix-based discrete event control of automated material handling systems.*" Proceedings of the 45th IEEE Conference on Decision & Control, 2006.

[42] NS2 Official Website:
http://nsnam.isi.edu/nsnam/index.php/Main_Page - 2008.

[43] OPNET Technologies, Inc. *http://www.opnet.com/* - 2009.

[44] Petri Nets World: http://www.petrinetz.de/ - 2009.

[45] Computer Tool for Coloured Petri Nets . CPN Tools: *http://wiki.daimi.au.dk/cpntools/cpntools.wiki* - 2007.

[46] The Reference Net Workshop. Renew : *http://www.renew.de/* - 2008.