PIPE v2.5: a Petri Net Tool for Performance Modeling

Pere Bonet, Catalina M. Lladó, Ramon Puigjaner
Departament de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
07071, Palma de Mallorca, Spain.
pere.bonet@gmail.com, cllado@uib.es, putxi@uib.es

and

William J. Knottenbelt
Department of Computing
Imperial College London
South Kensington Campus, SW7 2AZ, United Kingdom
wjk@doc.ic.ac.uk

Abstract

The Petri net modeling formalism allows for the convenient graphical visualization of system models, as well as the analysis of correctness and performance properties. Petri Nets theory has been widely used to implement a variety of modeling and evaluation tools. This paper surveys various of these tools and describes some recent extensions to a freely available and open-source platform-independent Petri net tool called PIPE. The extensions comprise: an increase in modeling power through the introduction of inhibitor arcs, a new analysis module for generating siphons and traps, some new interface features and various presentation improvements. We will illustrate system modeling and analysis with inhibitor arcs in a case study of a token ring network.

Keywords: Petri Nets, Performance Modeling, Performance Analysis, Modeling Tools

Resumen

El formalismo de modelado de redes de Petri permite la utilización de una conveniente visualización gráfica de los modelos de sistemas, así como el análisis de las propiedades de corrección y rendimiento. La teoria de las redes de Petri ha sido ampliamente utilizada para implementar herramientas de modelado. Este artículo estudia varias de estas herramientas y describe las extensiones recientes hechas a una herramienta de redes de Petri, *open-source* e independiente de la plataforma, llamada PIPE. Dichas extensiones comprenden: un incremento en la potencia de modelado a través de la introducción de arcos inhibidores, un nuevo módulo de análisis para la generación de sifones y lazos, nuevas características en la interfaz gráfica y varias mejoras en la presentación. El modelado de sistemas y el análisis con arcos inhibidores es ilustrado en un caso de estudio de una red token ring.

Palabras clave: Redes de Petri, Modelado de Rendimiento, Análisis de Rendimiento, Herramientas para el Modelado

1 Introduction

Any developer of computer or communication systems knows that the most important quality of a system is that it be functionally correct, i.e. that it exhibits certain behavioral, or *qualitative* properties. Once assured that the system behaves correctly, it is also important to ensure that the system meets certain performance-related (or *quantitative*) objectives. Indeed, while system users often take behavioral correctness for granted, it is the latter quantitative properties which often determine the success of one system over another [8].

Petri nets are a popular graphical modeling formalism that can help system designers ensure correctness and performance at design time. While they were originally developed for the study of qualitative properties of systems exhibiting concurrency and synchronization, temporal specifications were introduced about two decades ago in various forms to Petri Nets in order to also allow for quantitative analysis.

The focus of the present paper is on the Petri net tool PIPE (Platform Independent Petri Net Editor). PIPE is an open source, platform-independent tool for creating and analyzing Generalized Stochastic Petri Nets (GSPNs) – one type of Petri net that allows for temporal specification using immediate and exponential delays. PIPE is implemented entirely in Java to secure the platform independence and provides an elegant, easy-to-use graphical user interface that allows for the creation, saving, loading and analysis of Petri nets. PIPE was initially built at Imperial College London by means of different student projects. A new version with considerable improvements on different aspects of the tool functionality has now been implemented at the Universitat de les Illes Balears.

The rest of the paper is organized as follows: we begin with a review of Petri nets in section 2. Section 3 describes three tools available in the Petri Nets Tool Database [3] and compares them with PIPE. Section 4 outlines the PIPE while section 5 describes the additional features that have been added. Section 6 describes a case study showing how models can be built using PIPE and its new features. Finally, section 7 reports conclusions and intended future work.

2 Petri Nets Review

Petri Nets were introduced by Carl Adam Petri in the early 1960s [17]. One of its main attractions as a modeling formalism is how the basic aspects of concurrent systems are identified both conceptually and mathematically.

Structurally, a Petri Net (PN) is a directed bipartite graph comprising *places* and *transitions* [5]. Places, drawn as circles, model conditions or objects. Inside the places are *tokens*, drawn as black dots, which represent the specific value of a condition or object. A particular arrangement of the tokens across all the places is known as a *marking* or *state*. The system begins in some initial configuration known as the *initial marking*. Transitions, drawn as rectangles, are used to describe events that may modify the system state. Directed arcs specify the relation between local states and events in two ways: they indicate the conditions under which the event can occur, as well as the local state transformations induced by the event.

The arcs of the graph are classified (with respect to transitions) as *input* arcs (arrow-headed arcs from places to transitions), *output* arcs (arrow-headed arcs from transitions to places) and *inhibitor* arcs (circle-headed arcs from places to transitions). Multiple (input, output, or inhibitor) arcs between places and transitions are permitted and annotated with a number specifying their multiplicities.

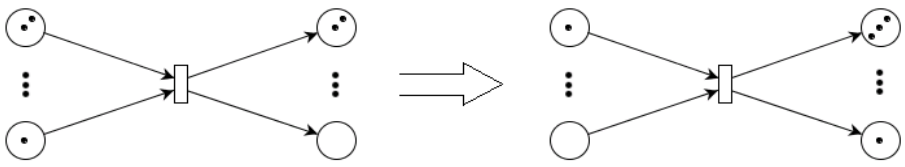

Figure 1: Firing of a transition

A transition is said to be *enabled* in a marking if each input place contains at least as many tokens as the multiplicity of the input arc and if each inhibitor place contains fewer tokens than the multiplicity of the inhibitor arc. Enabled transitions can *fire*, an atomic action which removes one token from all of its input places, and generates one token in each of its *output places*. Figure 1 shows the state of a Petri net before and after the firing of a transition. When arc weights larger that one are used, the number of tokens required in each input place for the transition enabling and the number of tokens generated in each output place by the transitions firing are determined by the weight of the arc connecting the place and the transition.

As an example, Figure 2 illustrates a PN that models two processes accessing a shared resource. The processes can be in one of three possible states: active (doing something that does not involve the shared resource), requesting the shared resource and accessing the shared resource. The shared resource can be idle or busy. The initial marking shows that in the initial state both processes are active and the shared resource is idle.

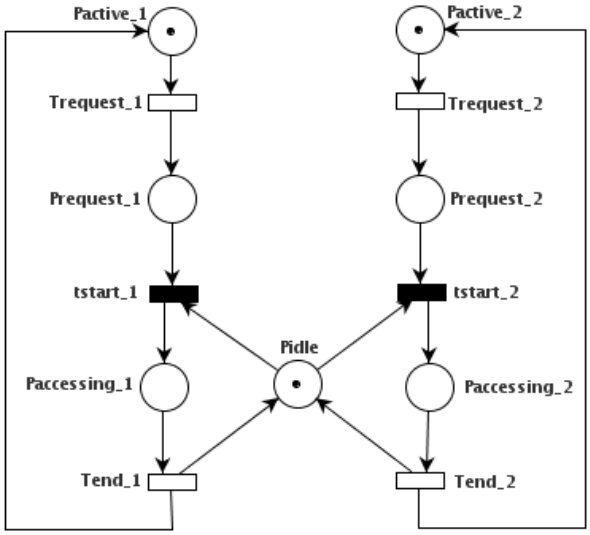

Figure 2: Producer-consumer GSPN model

Starting with the initial marking, transition occurrences (firings) yield new markings, that in turn give rise to further transition occurrences. Sequences of transition occurrences (also called *occurrence sequences*) are used to define *qualitative* (or behavioral) properties of PNs, such as *deadlock freedom* (no total deadlock is reachable), *liveness* (no partial deadlock is reachable), *boundedness* (on each place the number of tokens cannot grow in an unlimited way), and the existence of *home states* (markings that can always be reached from any reachable state). Occurrence sequences are normally represented as a *marking graph*, which is an arc-labelled directed graph with a distinguished initial vertex and edges labelled by transitions: the *vertices* are the reachable markings, the distinguished *initial vertex* is the initial marking, labelled edges are all triples (m, t, m') such that m and m' are reachable markings satisfying $m \xrightarrow{t} m'$.

2.1 Generalized Stochastic Petri Nets

We are interested in the so-called Generalized Stochastic Petri Nets (GSPN) which are characterized by 2 types of transitions:

- Stochastic transitions: associated with an exponentially distributed firing delay.
- Immediate transitions: associated with a null firing delay.

Formally, a GSPN can be defined as follows:

$$GSPN = (S, T, \Pi, I, O, H, M_0, W)$$

where

- S is a set of places
- T is a set of transitions, $S \cap T = \emptyset$
- $I, O, H : T \rightarrow N$ ($N = S \cup T$), are the input, output and inhibition functions
- $M_0 : S \longrightarrow I\!N$ is the initial marking
- $\Pi : T \longrightarrow I\!N$ is the priority function that associates the lower priorities to timed transitions and higher priorities to immediate transitions. Immediate transitions therefore always have priority over timed ones.
- $W : T \rightarrow R$ is a function that associates a real value to the transitions. $w(t)$ is:
  - a (possibly marking dependent) rate of a negative exponential distribution specifying the firing delay, when transition t is a timed transition (represented by a hollow rectangle).
  - a (possibly marking dependent) firing weight, when transition t is immediate (represented by a filled rectangle).

When a new marking is reached, if only timed transitions are enabled, this marking is *tangible*; if at least one immediate transition is enabled, the marking is *vanishing*.

The selection of which transition will fire is based on the priorities and weights. First, the set of transitions with highest priority is found and if it contains more than one enabled transition, the selection is based on the rates or weights of the transitions according to the expression:

$$P\{t\} = \frac{w(t)}{\sum_{t' \in E(M)} w(t')} \tag{1}$$

where $E(M)$ is the set of transitions enabled at the marking M i.e. the set of enabled transitions with highest priority.

2.2 Structural Analysis

For bounded PNs, the marking graph can be constructed to analyze certain behavioral aspects. However, even if the marking graph is finite, it is not always feasible to explicitly construct it because its size can explode exponentially (the so-called state space explosion problem).

Occurrences of transitions transform the token distribution of a net, but often respect some global properties of markings. For example, the total token count of a set of places remains unchanged if the pre-set and the post-set of the transitions contain the same number of places of this set. The notion of place and transition invariants formalize such invariant properties.

A vector $v \in \mathbb{Z}^n, v \neq 0$ is a place invariant of a Petri net with n places if $v^T M = v^T M_0$ for all reachable markings of the Petri net. The existence of a place invariant implies that the weighted token sum on a group of places remains constant for any evolution of the Petri net.

A vector $w \in \mathbb{Z}^m, w \neq 0$ is a transition invariant of an Petri net with m transitions if $Cw = 0$ where C is the incidence matrix of the net. A transition invariant describes a group of transitions that may be fired without affecting the marking of the net.

Next, we consider sets of places, S that never gain a token once none of their places is marked. A *siphon* is a set of places S satisfying $\cdot S \subseteq S\cdot$. A siphon is *marked* by a marking m if at least one of its places is marked by m. Given a PN with a siphon S, if S is not marked at the initial marking, then S is not marked at any reachable marking.

Similarly to siphons, we consider sets of places that never lose all tokens once at least one of their places is marked. A sufficient condition is that each transition that removes at least one token from this set also adds a token. This condition is formalized by the notion of trap:

A *trap* is a set T of places satisfying $T\cdot \subseteq \cdot T$. A trap is marked by a marking m if at least one of its places is marked at m. Given a PN with a trap T, if T is marked at the initial marking then it is marked at every marking.

Place and transition invariants as well as siphons and traps can be used as an inexpensive way to check for certain qualitative properties of systems such as deadlock freedom, liveness and boundedness.

2.3 Performance Analysis

The analysis of a GSPN model requires the solution of a system of linear equations comprising as many equations as the number of reachable *tangible* markings. The steady-state solution of the model is obtained by solving the system of linear equations:

$$\pi Q = 0$$
$$\sum_{M \in RS} \pi[M] = 1$$

Q is the infinitesimal generator matrix, whose elements outside the main diagonal are the rates of the exponential distributions associated with the transitions from state to state, while the elements on the main diagonal make the sum of the elements of each row equal to zero.

π is the equilibrium probability mass function (pmf) over the reachable markings; we write $\pi[M]$ for the steady-state probability of a given marking M.

The transient solution of the model is obtained solving the set of differential equations:

$$\frac{d\pi(t)}{dt} = Q\pi(t) \tag{2}$$

where $\pi(t)[M]$ is the probability of the system being in state M at time t.

Several kinds of aggregate results are easily obtained from the steady-state distributions over reachable markings. Some of the more commonly computed aggregated performance measures are [19, 5]:

- The pmf of the number of tokens at steady-state in a place, say p can be obtained by computing the individual probabilities in the pmf as probabilities of the event "place p contains k tokens".
- The average number of tokens in a place can be computed from the pmf of tokens in that place.
- The frequency of the firing of a transition (throughput), can be computed as the weighted sum of the transition firing rate, i.e.:

$$f_{t_j} = \sum_{M_i \in EN_j} w(t_j, M_i)\pi_i \tag{3}$$

3 Petri Nets Tools

Petri nets theory has been widely used to implement a variety of modeling and evaluation tools; most of them can be found at the so-called Petri Nets Tool Database [3]. Some of them are also surveyed in [18] as part of the standardization effort that was carried at the beginning of this decade for defining a common interchange format for Petri nets models. One of the outcomes of this effort is The Petri Net Markup Language (PNML) [2, 9], an XML-based interchange format for Petri nets that has been used for some tools. However, this format is not in fact, agreed as standard.

Unfortunately, many of the tools described on the database or in literature are not longer maintained or available. Moreover, many of them have problems including: installation problems, serious functionality

problems that hamper usability, lack of support for PNML and a lack of analysis power necessary to conduct both qualitative and quantitative analysis on (at least) GSPNs. Still fewer are free of charge and/or open source. Therefore, when looking for a Petri net tool to be used in academia, for example to give a course in performance modeling for concurrent systems, finding a suitable one is not an easy job. We hope this brief survey presented in subsection 3.1 and the tool PIPE presented in this paper can help in such a task.

3.1 Petri Nets Tools for Performance Evaluation

This section surveys the following tools:

- GreatSPN 2.0 [1, 11] (GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets): a software package for the modeling, validation, and performance evaluation of distributed systems using GSPNs and their colored extension: Stochastic Well-formed Nets. Unfortunately, only source codes are provided to compile this tool under Unix or Linux without giving any help about the needed packages for its compilation. As a result, it is quite hard to get the tool compiled and working. In addition, some error messages are cryptic; for example, when the steady-state analysis does not finish for some reason (possibly for lack of resources) the error obtained is *Error in child termination: Bad file descriptor*.
- Sharpe [20] (Symbolic Hierarchical Automated Reliability and Performance Evaluator): a tool for specifying and analyzing performance, reliability and performability models that allows for different model types. These include queueing networks, Markov and semi-Markov reward models as well stochastic Petri nets. Output measures of a model can be used as parameters of other models which facilitates the hierarchical combination of different model types. The tool includes a GUI with reduced functionality which does not deal with Petri nets.
- TimeNet 4.0 [4, 12, 21] (TIME Net Evaluation Tool): a tool for the modeling and analysis of stochastic Petri nets with non-exponentially distributed firing times. This version (4.0) contains a new JAVA-based user interface.
- PIPE 2.5, described in more detail in sections 4 and 5.

All these tools have the following characteristics in common: free of charge for academia, somehow maintained, Graphical User Interface (GUI), modeling of GSPNs (at least), parameter specification, inhibitor arcs, structural analysis, GSPN steady-state analysis and simulation: giving all of them at least the subsequent performance results:

- Steady-state probability distribution of tangible states.
- Average number of tokens at each place.
- Throughput of transitions.

Other important features that are not shared between these tools are summarized in Table 1.

4 PIPE background

PIPE is an open source, platform independent tool for creating and analyzing Petri nets including GSPNs. It is implemented entirely in Java to secure the platform independence and provides an elegant, easy-to-use graphical user interface that allows creating, saving and loading of Petri nets conforming to the PNML interchange format. PIPE also offers a full suite of analysis modules to check behavioral properties, produce performance statistics, and some less common features such as PNs comparison and classification.

PIPE began life in 2002/3 as an MSc Group Project in the Department of Computing at Imperial College London. With up to 4 successive versions (all of them implemented by students at Imperial College London), the GUI has been highly improved, new features that increase the modeling power and new analysis modules have been added, result in a quite complete, mature and efficient tool.

Features	GreatSPN 2.0	Sharpe	TimeNet 4.0	PIPE 2.5
Operating system	Unix	Any	Linux/Windows	Any
PNML based	No	No	No	Yes
GUI				
Undo	No	No	Yes[1]	Yes
Redo	No	No	No	Yes
Copy/Paste	Yes	No	Yes	Yes
Arc weighs	Yes	Yes	No	Yes
Features				
Capacity Limitation	Yes	No	No	Yes
Priorities	Yes	No	Yes	Yes
Server Semantics	Yes	No	Yes	No
Token Game				
Step by Step	Yes	No	Yes	Yes
Backwards	No	No	No	Yes
Continuous	Yes	No	Yes	Yes
Structural Analysis				
Net Comparison	No	No	No	Yes
Place Invariants	Yes	No	Yes	Yes
Transition Invariants	Yes	No	No	Yes
Siphons and Traps	Yes	No	Yes	Yes
Qualitative Properties	All	None	None	All[2]
Analysis				
Transient Analysis	Yes	No	Yes	No
Transient Simulation	Yes	No	Yes	No

Table 1: Petri Nets tool comparison

In the subsequent subsections the main features of PIPE2 are described. More information on the improvements of each version can be found at [10, 7, 6, 15].

4.1 The Graphical User Interface

PIPE was designed with the objective of providing an intuitive, user-friendly tool for editing Petri nets in a easy, fast and efficient way. Anyone familiar with the standard drawing UI can pick up and use PIPE without application specific knowledge. The editor uses standard representation for the different elements that constitute a Petri net as described in section 2.

Following, the remarkable features of the GUI are described:

- It conforms the XML/PNML standard so it could open and work with existing PNML Petri nets. In addition, PNML annotations can also be added, so the user can include text to explain detail of the created models.
- It provides a multiple document interface so that one can work with multiple nets at the same time, each of them located on a tabbed pane.
- Users are able to perform tasks using a menu bar, a toolbar and mouse actions. Moreover, some keyboard shortcuts are also available to allow for quicker actions.
- Nets can be printed or exported into two graphical formats: Postscript and PNG.
- A zoom functionality is provided to grant the user greater flexibility and much easier use when working with large nets.

4.2 Animation mode/Token game

PIPE offers an animator so that the user can manually experiment with the token game, firing any of the enabled transitions at each state. The set of enabled transitions is highlighted and the user chooses which

[1] It is implemented but does not always work well, for example after having pasted a selection the *Undo* does not work
[2] For liveness properties uses a necessary condition but not sufficient

one must be fired. Animation history is recorded, i.e. all the fired transitions can be seen on the side of the screen, so from the current state the animation can be stepped forwards or backwards. The automatic execution of a random transition in animation mode is also possible; for this the user specifies the firing delay and the number of firings.

4.3 Analysis Modules

PIPE offers a set of modules to carry out different types of qualitative and quantitative analysis. These set of available modules can easily be increased, provided that the defined interface is followed. The available modules in PIPE are the following:

- Classification: based on the connectivity between places and transitions, this module classifies a Petri net into one or more of the following types: State Machine, Marked Graph, FC-Nets, EFC-Nets, SPL-Nets, ESPL Nets.
- Comparison: this module compares two Petri nets based on their PNML files. It confirms whether they are (functionally) the same; otherwise, it works out the differences between them.
- DNAmaca interface: through this module, PIPE is able to interface with DNAmaca, a Markov chain specification, generation and solution tool, which can be used to perform analysis of GSPNs. It is capable of performing passage time analysis on both semi-Markovian and Markovian models, although this module focuses on the Markovian solver. The results produced by this module are passage time analysis statistics for the current active net, which represent the distribution of how long it takes for a system to go from the source state to the target state (both provided by the user).
- GSPN analysis: this module calculates the average number of tokens on a place, the token probability density and the throughput of timed transitions by exploring the state space of the given Petri net and determining the steady state solution of the model.
- Invariant analysis: this module computes the place invariant and transition invariant vectors accurately and efficiently. These vectors (called net-invariants) are not hard to compute since the complexity of the calculation depends only on the number of places and transitions in the net and not on the size of the reachability set. It also provides the marking equations and information about boundedness and liveness.
- Incidence & marking: this module displays the forward, backward and combined incidence matrices and the marking matrix and the set of enabled transitions. It enables the user to feel more confident about the validity of the Petri net because the intermediate results used to generate the more advanced results can be viewed.
- Reachability graph: this module provides a visual representation of all the possible firing sequences for the given Petri net, as described in section 2. It also gives information about boundedness, safeness and deadlock-free properties.
- Simulation: by means of simulation, this module computes the average number of tokens per place along with the 95% confidence interval for each place in the net. A full analytical approach to a problem often demands a huge amount of resources, while simulation can provide an alternative method to overcome performance issues. This module runs without any input from the user and operates in conjunction with the analysis modules so that performance results of the simulated system can be computed.
- State space analysis: this module builds a tree of all the reachable markings which is used to determine the qualitative properties of the given Petri net: boundedness, deadlock-free, and safeness. It also provides the shortest path to deadlock in the case that there exists one.

5 PIPE 2.5

We have upgraded PIPE with yet another version (PIPE version 2.5). The improvements introduced by this new version are detailed in the following subsections.

5.1 Increase of the Modeling Power

PIPE's modeling power has been increased with the addition of inhibitor arcs, capacity restrictions to places and priorities for immediate transitions.

- Inhibitor arcs: As shown by Figure 3, being able to use inhibitor arcs makes the modeling of certain types of structures much easier. Moreover, with inhibitor arcs, a Petri net has the ability of *zero testing* which gives Petri nets the power to simulate a Turing Machine [16].

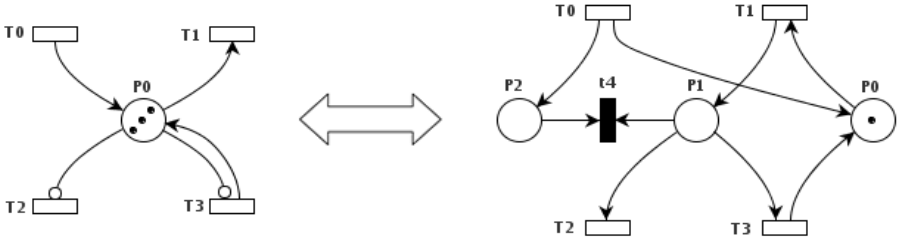

Figure 3: These two portions of nets are equivalent

- Capacity restriction of places: The number of tokens allowed in a place with capacity restriction is limited to that capacity. As a consequence the enabling rule for the transitions changes (with respect a enabling rule for Petri nets without this restriction) in the following way: a transition with an output place with restricted capacity is only enabled if the standard conditions apply (as described in section 2) and if the number of tokens after the firing of the transition does not exceed the value of the place capacity restriction.
- Immediate transition's priorities: implementing the priorities as defined in section 2.

Clearly, the addition of these features implies non trivial modifications in the tool implementation as for example, the creation of new data structures to allow for inhibitor arcs, the update of the algorithm that determines if a transition is enabled (due to the addition of inhibitor arcs, capacity restriction and priorities), the saving/loading of the PNML files and the displaying of the new type of elements.

5.2 Analysis Modules

Taking advantage of the PIPE's modular structure, a new analysis module has been developed. This module generates the minimal siphons (sets of places that never gain a token once none of their places is marked [19]) and traps (sets of places that never lose all tokens once at least one of their places is marked [19]) for a given net [14]. Traps and siphons can be used for the structural analysis of net systems e.g. to check liveness and reachability. See section 2 for more details.

Additionally, the presentation format for the results of the invariant analysis module has been modified since a lack of information on them at the previous version was making difficult its interpretation.

5.3 Improving the Graphical Editor

Many of the new features of PIPE 2.5 are focused on the graphical editor (shown in Figure 4). Two key elements of any graphical editor are the cut/copy/paste functionalities, and the undo/redo capabilities. These features were missing in previous versions of PIPE and so have been added to the tool. Now, the user can select individual net elements, or entire sections of the net, and copy them into the same net or into another net (placed in another tab). Also, every change made can now be undone and redone (with a limit of 50 edits).

Another new feature of PIPE 2.5 is that allows the possibility of specifying the initial marking and transition rates as parameters that are defined for the whole model. With them, users can easily work with different configurations of these input parameters.

Labels have also been improved so that they can display the attributes of the net element which they are related to (e.g. a transition's priority and rate). Place and transition labels can also be moved independently of the elements they relate. This is especially useful when dealing with complex nets since labels can be involuntarily hidden. To address this issue, labels can now be placed (in a relative position) wherever the user decides (which was not possible with previous versions).

Other improvements are summarized as follows:

- A new *fast editing mode* has been implemented which allows the user to link places and transitions to build the net structure much quicker.
- The mouse wheel, if present, has been enabled to execute various actions in a context dependent way: e.g. to increase/decrease the marking of a place, to edit the weight of an arc or to rotate a transition.
- For the bidirectional arcs, the user can now choose to draw them as a double-headed arc or as two separate arcs.
- New dialogs accessible via the context menu have been added. They permit easy modifications of the element's attributes.

6 Case Study

To illustrate the capabilities of PIPE, we present a case study where a Token-Ring network is modelled as a GSPN [13]. Consider a unidirectional ring network having a fixed number of ports, labelled $1, 2, \ldots, N$ in the direction of the signal propagation. At each port, message packets arrive according to a random process. A distinguished bit pattern, called a *ring token*, circulates around the ring from one port to the next. When a port observes the ring token and has a packet queued for transmission, the port converts the ring token to another distinguished bit pattern called a *connector* and transmits the packet followed by the ring token; the ring token continues to propagate if the port has no packet queued for transmission. Conceptually, the port *removes the token* from the ring at the start of the transmission, *holds the token* while the transmission is underway, and *releases the token* back onto the ring at the end of the transmission. By destroying the connector the port *removes* the transmitted packet when it returns around the ring. For simplicity, we assume that at most one packet is awaiting for transmission at any time, at any particular port.

This system can be specified as a GSPN with inhibitor arcs, as shown in Figure 4 for $N = 2$. Place P8 (P9) contains a token if and only if the port observes the ring token. If the port does not have a packet to send (there is no token in P0 (P1)) it starts the propagation of the ring token (transition T7 (T5)) to the next port. Otherwise it starts the transmission of the packet (transition T6 (T4)). Duration of transmission is modelled by T2 (T4). Once the packet has been sent, the port starts the propagation of the ring token to the next port (modelled by T8 (T9)).

The use of inhibitors arcs, now possible with PIPE, makes the graphical representation of the token ring much simpler than it would be if those had to be represented with its equivalent as shown in Figure 3. The use of parameters is also illustrated by this example.

The performance analysis carried out with the PIPE GSPN analysis module gives the following results (among others): the average number of tokens per place (see Table 2), the token probability density (see Table 3) and the throughput of timed transitions (shown in Table 4). Results are somehow monotonous because of the simplicity of the network. However, using a more realistic model (for example with more ports, no restriction etc.), results similar to these could be used to obtain average waiting times at the ports for different values of the arrival rate parameter.

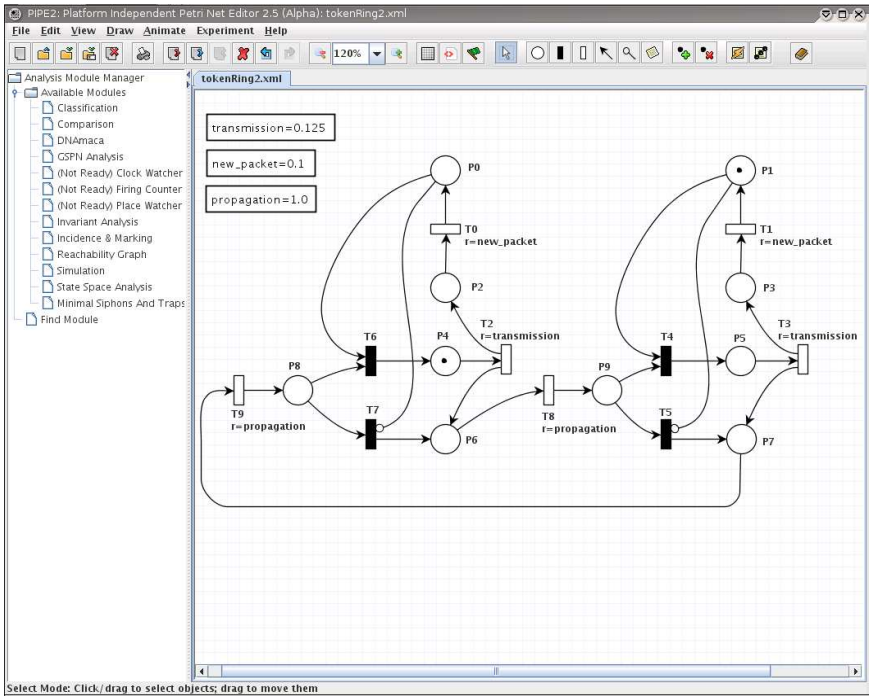

Figure 4: Two ports Token-Ring

Place	Number of Tokens
P0	0.54054
P1	0.54054
P2	0.22973
P3	0.22973
P4	0.22973
P5	0.22973
P6	0.22973
P7	0.22973
P8	0
P9	0

Table 2: Average number of tokens on a place

Place	$\mu = 0$	$\mu = 1$
P0	0.45946	0.54054
P1	0.45946	0.54054
P2	0.77027	0.22973
P3	0.77027	0.22973
P4	0.77027	0.22973
P5	0.77027	0.22973
P6	0.77027	0.22973
P7	0.77027	0.22973
P8	1	0
P9	1	0

Table 3: Token probability density

T	Throughput
T0	0.04276
T1	0.04276
T2	0.04276
T3	0.04276
T8	0.15791
T9	0.15791

Table 4: Throughput of timed transitions

7 Conclusions

Petri nets are a popular graphical modeling formalism that can help system designers ensure correctness and performance at design time. Petri nets theory has been widely used to implement a variety of modeling and evaluation tools. In this paper we have surveyed some of them and compared those to the PIPE open source Petri net editor. We have also described some recent improvements made to PIPE, including modeling power enhancements, additional analysis functionality and interface improvements. Future work includes allowing for the specification of server semantics for timed transitions and integrating PIPE with a GRID-enabled analysis backend and more powerful and natural query specification mechanism.

References

[1] GreatSPN 2.0. GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. `http://www.di.unito.it/~greatspn/`.

[2] Petri Net Markup Language. http://www2.informatik.hu-berlin.de/top/pnml/about.html.

[3] Petri Nets Tool Database. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html.

[4] TimeNET 4.0. TIMEd Net Evaluation Tool. http://pdv.cs.tu-berlin.de/~timenet/.

[5] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. *Modelling With Generalized Stochastic Petri Nets*. Wiley, 1995.

[6] Akharware, N. PIPE2: Platform Independent Petri Net Editor. http://pipe2.sourceforge.net/documents/PIPE2-Report-20050814.pdf, 2005.

[7] Barnwell, T., Camacho, M., Cook, M., Gready, M., Kyme, P., and Tsouchlaris, M. Final report. http://pipe2.sourceforge.net/documents/PIPE2FinalReport.pdf, 2004.

[8] Bause, F., and Kritzinger, P. *Stochastic Petri Nets*, 2nd ed. Vieweg, 2002.

[9] Billington, J., et al. The Petri Net Markup Language: Concepts, Technology, and Tools. http://www2.informatik.hu-berlin.de/top/pnml/download/about/PNML_CTT.pdf, March 2003.

[10] Bloom, J., Clark, C., Clifford, C., Duncan, A., Khan, H., and Papantoniou, M. Platform Independent Petri-net Editor final report. http://pipe2.sourceforge.net/documents/PIPE-Report.pdf, 2003.

[11] Chiola, G., Franceschinis, G., Gaeta, R., and Ribaudo, M. GreatSPN 1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation, special issue on Performance Modeling Tools 24*, 1-2 (November 1995), 47–68.

[12] German, R., Kelling, C., Zimmermann, A., and Hommel, G. TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets. *Performance Evaluation 24*, 1-2 (November 1995), 69–87.

[13] Haas, P. J. *Stochastic Petri Nets: Modelling, Stability, Simulation*. Springer-Verlag, New York, 2002.

[14] Jeng, M. D., and Peng, M. Y. Generating Minimal Siphons and Traps for Petri Nets. In *Proceedings of IEEE International conference on Systems, Man, and Cybernetics* (Beijing, China, October 1996), pp. 2996–2999.

[15] Kimber, T., Kirby, B., Master, T., and Worthington, M. Petri nets group project final report. http://pipe2.sourceforge.net/documents/PIPE2-Report-20070327.pdf, 2007.

[16] Peterson, J. L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[17] Petri, C. Communication With Automata. Tech. Rep. RADC- TR-65-377, Rome Air Dev. Center. New York, 1966.

[18] R., B., Buchs, D., Buffo, M., Kordon, F., and Sy, O. Questionnaire for a Taxonomy of Petri Net Dialects. http://www-src.lip6.fr/homepages/Fabrice.Kordon/PN_STD_WWW/pdf_result.pdf, May 2000.

[19] Reisig, W., and Rozenberg, G. E. *Lectures on Petri Nets I: Basic Models*. Springer, 1998.

[20] Trivedi, K., and Hirel, C. SHARPE: Symbolic Hierarchical Automated Reliability and Performance Evaluator. http://www.ee.duke.edu/~chirel/IRISA/sharpeGui.html.

[21] Zimmermann, A., Knoke, M., Huck, A., and Hommel, G. Towards Version 4.0 of TimeNET. In *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems, MMB 2006* (March 2006), pp. 477–480.