Hierarchically constructed Petri-nets and product-forms

Peter G. Harrison
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ, UK

pgh@doc.ic.ac.uk

Catalina M. Lladó
Departament de Ciències, Matemàtiques i Informàtica
Universitat de les Illes Balears
07071 Palma de Mallorca, Spain

cllado@uib.cat

ABSTRACT

Stochastic Petri nets (SPNs) provide a convenient, diagrammatic description of concurrent systems, such as computer and communication networks, and can represent quantitative (or performance) aspects such as mean response times and probability of failure. Such models can be supported by performance modelling interchange formats (PMIFs), facilitating sharing and model interoperability. We propose a hierarchical method for constructing a large class of Petri nets, which preserves efficient product-form solutions when they exist. This scalable approach greatly improves the efficiency of finding steady state probabilities in a wide range of SPNs, making much larger SPNs feasible. An existing PMIF is extended by including a new type of node that describes a particular type of small Petri net, called a "building block", the synchronisation primitives for which can be used to specify task-spawning and task-gathering, whilst retaining product-form solutions under specified conditions. When there is no product-form, the whole network is translated into a Petri net and solved directly – either by a Markov chain solver or by simulation. The extended PMIF and the proposed methodology are applied to a model of a computer system with RAID storage.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance Analysis and Design Aids; C.4 [Performance of Systems]: Modeling Techniques.

General Terms

Performance engineering, Markov processes, model interoperability

Keywords

Petri nets, product-forms, performance modelling interchange formats, RAID systems.

1. INTRODUCTION

The Stochastic Petri-net (SPN) is an expressive, graphical formalism for performance modelling that is generally specified at a high level, in terms of workflows and constraints, perhaps via a higher-level language such as UML. However, solving for performance metrics generally requires mapping onto an underlying continuous time Markov chain (CTMC), which, in turn, is solved by direct methods in an essentially "brute force" approach. This procedure is computationally expensive – in terms of both computation time and storage – and so limited for practical use. Worse still, many product-form models that are known in other formalisms, like queueing networks, become obscured in their Petri net form and hence are also solved inefficiently by direct solution of their CTMC, the very thing that product-forms sought to circumvent in the first place. The choice facing the system modeller is this: to use either a Petri-net diagram (or higher level abstraction like UML) that he or she can understand relatively easily, but which has an inefficient solution, or derive a set of mathematical equations that are conducive to transformation into an efficient solution; in particular, a product-form. We show how to achieve the best of both worlds: a way of constructing hierarchical SPNs in terms of primitive "building blocks" (BBs) – themselves small, "flat" SPNs – that are both easy to read and have solutions that are as efficient as those obtained from the mathematical equations of, say, queueing network analysis. Under appropriate conditions, product-forms for the equilibrium probabilities of recursive compositions of the BBs are obtained using the Reversed Compound Agent Theorem (RCAT) of [9]. Moreover, we integrate the hierarchical class of SPNs thus defined into a generalised performance modelling interchange format (GPMIF) to provide a powerful, general and rigorous route to product-forms in a large and significant class of stochastic models. This is illustrated by a case study in the domain of RAID storage systems.

Performance modelling interchange formats (PMIFs) are XML schemata that facilitate seamless sharing of performance models amongst appropriate software tools [19]. It is not necessary for these tools to know about the capabilities of other tools or their internal data formats. The only requirement for import/export is that they must either support the PMIF itself or else provide an interface that reads/writes model specifications from/to a file. Interchange formats have been defined for specific formalisms such as queueing network models (QNM), software performance models (S-PMIF) and layered queueing networks (LQN).

A generalised PMIF (called GPMIF) extended the original, QNM-based PMIF of [19] to incorporate negative customers and fixed point models in such a way that product-form

VALUETOOLS 2011, May 16-20, Paris, France

DOI 10.4108/icst.valuetools.2011.245746

solutions, when they existed, would still be found [12, 8]. In the present contribution, we further extend the GPMIF to hybrid models by including a new type of node that describes the SPN "building blocks" referred to above. In fact the simplest building block, BB-1, consists of a single Petri net place having a single input transition and a single output transition, with no capacity constraint, which is equivalent to an M/M/1 queue. Hence all queueing networks with no capacity constraints fall into a subset of the new formalism; indeed, multi-class queueing networks may also be included and further extension to incorporate negative customers is also possible [8]. Higher-order building blocks, BB-n, have n places with pairs of input and output transitions that connect to the same subset of k places ($1 \leq k \leq n$). The BBs are defined in the enlarged GPMIF schema and models that use them can be implemented, in general, by simulation or direct solution of the underlying Markov chain, or via a product-form solution when one exists. For completeness, the extension to the PMIF we have implemented incorporates all simple Petri nets, not only those constructed from BBs.

The derivation of product-forms by RCAT imposes conditions in the extended GPMIF that are very similar to those required of queueing networks [9]. As a result, complex systems can be solved efficiently and mechanically at equilibrium; for a wide range of parameterisations in closed systems and under certain conditions on the rates in open systems. We focus on fork-join operations and solve a RAID-like model by two solution methods for the same specification: via the Petri net analyser PIPE (Platform Independent Petri net Editor) and a product-form solution [1, 2].

Related work is discussed in the next section, before section 3 provides background material on concepts related to the present paper: PMIFs, queues and Petri nets, including the definition of the new BBs. Furthermore, the construction of a product-form solution is illustrated by a simple example comprising a BB-2 building block connected to three queues (i.e. BB-1s). Section 4 defines the further extended GPMIF by showing how an XML specification for BBs can be integrated into the current GPMIF schema (already an extension of the original PMIF schema, as noted above). The full schema, which applies to more general Petri nets, is too large to include but is available at `dmi.uib.es/~cllado/mifs/`. Section 5 describes two solution methods: product-form solution and simulation. It also elaborates the simple example as an illustrative exemplar for more complex applications of the extended schema. An abstract case study is presented in section 6 which is solved both as a product-form for the equilibrium state probabilities and by simulation via translation into PIPE. Both open and closed, product-form and non-product-form cases are considered. The paper concludes in section 7.

2. RELATED WORK

2.1 Multi-formalism performance models

Extensive work has been done on multi-formalism performance models. Sharpe [20] was one of the earliest experiences, where submodels based on different formalisms interact by exchanging probability distributions to obtain a global result. More recently, Mobius, OsMoSys (Object-based multi-formaliSm MOdeling of SYStems) and SIMTHESys (Structured Infrastructure for Multiformalism modeling and Testing of Heterogeneous formalisms and Extensions for SYSTems) have followed a similar path [17, 21, 14]. All of them aim to provide a methodology and tool support for multi-formalism models' design and evaluation, and consider model composition and multiple solution methods. All of the approaches use submodel composition to support interaction amongst formalisms, but with different premises. In Mobius, submodels interact by sharing state variables and by superposing events between submodels. More specifically, in order for a formalism to be compatible with the Mobius framework, it must be possible to translate any of its models into an equivalent model that uses Mobius framework components. Because all models are transformed into framework components, they and their solution techniques in the framework are able to interact with each other. The framework is also extensible, allowing new formalisms and solvers to be added with little impact on existing ones, since new formalisms and solvers also communicate using framework components.

In OsMoSys, interactions between formalisms are defined by using operators that formally describe the semantics of information exchange between submodels. Finally, in SIMTHESys multi-formalism cooperation is achieved by allowing heterogeneous models to be considered. This can be implemented by proper composition formalisms that define hybrid elements, the behaviours of which are specifically designed for interaction with elements of different formalisms, and encapsulate in their behaviours an interaction logic.

What we propose instead is a general performance model interchange format that allows for specifying hybrid networks. We show how these hybrid networks can be solved in different ways and with tools depending on the specific models. In our case study, the model can be solved by RCAT as a product-form but alternatively it can be completely transformed into a pure Petri net and solved by PIPE.

2.2 Product-form Petri nets

A product-form result for Petri nets was obtained by Boucherie in [6], where it was found that two SPNs, whose stationary solutions are known, can be composed under a strict blocking discipline to yield a product-form solution. More generally, but non-incrementally, Henderson *et al.* obtained a method to determine product-forms in fully specified SPNs that satisfy certain structural conditions [13, 7]. This involves solving a certain set of linear equations on the whole net, akin to the traffic equations of queueing networks, together with a condition on the rank of a matrix, the elements of which depend on the net's transition rates.

More recently, in [16], it was shown that a class of SPNs with a rate-independent product-form condition is equivalent to the class of chemical reaction networks that satisfies the so-called *deficiency zero* property. It was further proved that a state-machine possessing the deficiency zero property is equivalent to a Jackson network.

3. PMIFS, QUEUES AND PETRI NETS

PMIF 2.0 is a common representation for system performance model data that can be used to move models among modeling tools that use a QNM paradigm [18]. GPMIF (Generalised PMIF) allows, in addition, the specification of

non-standard queueing networks, such as G-networks, and certain fixpoint solutions [11, 8][1].

A natural, orthogonal direction to extend GPMIF is to find non-queueing building blocks that can be defined as primitive service centres with associated workloads that interact with the existing components. We look to Petri nets for such building blocks because of their ease of expressing features found in real networks and their widespread use. To this end, consider a building block that consists of a set of places $P_1, \dots P_N$, a set $\mathcal{T}_I$ of input transitions whose input vectors are null (i.e. $\mathbf{0} = (0, \dots, 0)$), and a set $\mathcal{T}_O$ of output transitions whose output vectors are null[2]. All the arcs have multiplicity 1.

Definition 1 (Building block (BB) [2]). *Given an ordinary (connected) SPN S with set of transitions $\mathcal{T}$ and set of N places $\mathcal{P}$, then S is a* building block *if it satisfies the following conditions:*

1. *For all $T \in \mathcal{T}$ then either $\mathbf{O}(T) = \mathbf{0}$ or $\mathbf{I}(T) = \mathbf{0}$. In the former case we say that $T \in \mathcal{T}_O$ is an* output transition *while in the latter we say that $T \in \mathcal{T}_I$ is an* input transition. *Note that $\mathcal{T} = \mathcal{T}_I \cup \mathcal{T}_O$ and $\mathcal{T}_I \cap \mathcal{T}_O = \emptyset$, where $\mathcal{T}_I$ is the set of input transitions and $\mathcal{T}_O$ is the set of output transitions.*
2. *For each $T \in \mathcal{T}_I$, there exists $T' \in \mathcal{T}_O$ such that $\mathbf{O}(T) = \mathbf{I}(T')$ and vice versa.*
3. *Given two places $P_i, P_j \in \mathcal{P}$, $1 \le i, j \le N$, there exists a transition $T \in \mathcal{T}$ such that the components i and j of $\mathbf{I}(T)$ or of $\mathbf{O}(T)$ are non-zero.*

Condition 1 requires that all the transitions are either input or output transitions, while Condition 2 states that if there exists an input transition T_y feeding a subset of places y, then there must be a corresponding output transition T'_y that consumes the tokens from the same subset; i.e. for each input transition T_y there must exist an output transition T'_y whose input vector is equal to the output vector of T_y. Finally, Condition 3 simply requires the SPN to be connected.

Figure 1 illustrates an example of a BB consisting of 3 places $\mathcal{P} = \{P_1, P_2, P_3\}$, 3 input transitions $\mathcal{T}_I = \{T_3, T_{23}, T_{12}\}$ and 3 output transitions $\mathcal{T}_O = \{T'_3, T'_{23}, T'_{12}\}$.

Note that if two or more input (output) transitions have the same output (input) vector, we can fuse them in one transition whose rate is the sum of the rates of the original transitions. Therefore, without loss of generality, we assume that all the input (output) transitions have different output (input) vectors.

Finally, to simplify the notation, we use T_y (T'_y) to denote an input (output) transition, where y is the set of place-indices of the non-zero components in the output (input) vector of T_y (T'_y). For instance, transition T_{23} (T'_{23}) in the net of Figure 1 is the transition that produces (consumes) the tokens in P_2 and P_3.

[1]Later, GPMIF was further generalised to accommodate the extended RCAT, whilst maintaining compatibility with its previous versions [10].

[2]The components of an input/output vector specify the number of tokens taken from / added to the corresponding places. Thus a null vector implies tokens arrive/depart externally.

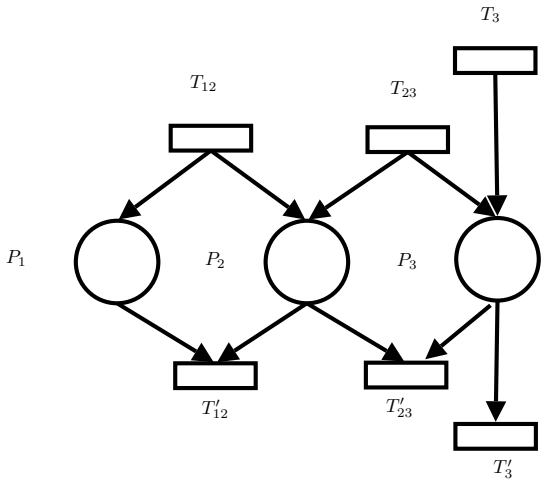

Figure 1: A 3-place building block.

Before giving the product-form for a general BB, we first consider a simple one, with just two places, depicted in Figure 2. We use the following conventions: transitions T_y, with null input vector, are always enabled; we call them *input transitions* and denote the set of all such T_y by $\mathcal{T}_I$. Transitions T'_y, with null output vector, are called *output transitions* and the set of all T'_y is denoted by $\mathcal{T}_O$. The subscript y is the set of indices of the output/input places for the input/output transition T_y/T'_y. The rates for this SPN are written $\chi_y = \lambda_y$ and $\chi'_y = \mu_y$ where $y = \{1\}, \{2\}, \{1, 2\}$.

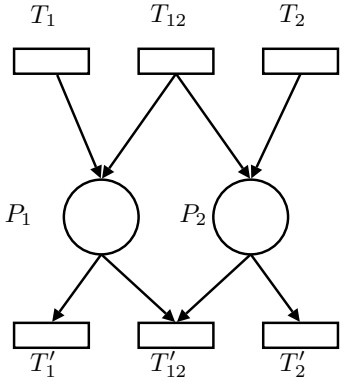

Figure 2: Two-place building block model.

Let P^1, P^2 be the Markov processes whose states represent the number of tokens in the places P_1, P_2, respectively. We use t_y to denote the action (name) associated with transition T_y, and t'_y similarly for T'_y.

3.1 Simple example

To illustrate, we consider the Markov process defined by the composition of the BB-2 node defined above with three M/M/1 queues. The outputs of the BB-2 node are connected to the inputs of the queues and the outputs of the

queues are connected to the three inputs of the BB-2 node; see Figure 3. As drawn, this network is closed, with a fixed population, and has product-form equilibrium joint probabilities for the numbers of tokens in each of the five places. If all the BB-2 outputs instead randomly choose an input to one of the M/M/1 queues with fixed probability (one third in the figure), a product-form still exists. If an additional input transition were now added as input to any of the places, the network would become open and have a product-form only under a certain condition on the rates. These product forms are examined by RCAT in section 5.1.1.

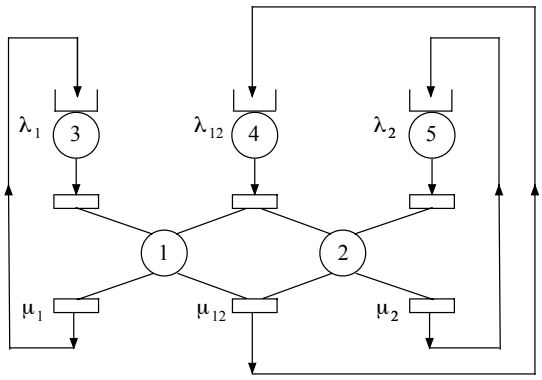

Figure 3: Simple closed hybrid network.

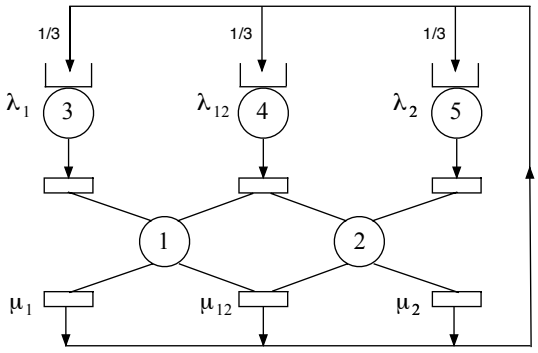

Figure 4: Closed network with random choice.

4. BB COMPONENTS IN GPMIF

This section describes the Petri net building blocks' specification in an extended GPMIF schema. Keeping the model interchange format compatible with PMIF 2.0 [19] and GPMIF [12], we added a new element *PNBuldingBlock* to GPMIF, which can be any GSPN, specified in PNML (Petri Net Markup Language). A partial view of the resulting schema showing the new elements is drawn in Figures 5 and 6.

4.1 Central server system with RAID

The following excerpt shows part of the BBs specification, based on the above schema, for the RAID case study of section 6.1. Specification of Nodes and Workloads is as specified in PMIF, see [19].

```
<PNBuildingBloc Name=''RAID''>
 <net id="Net-One" type="P/T net">
 <place id="RAID-1">
```

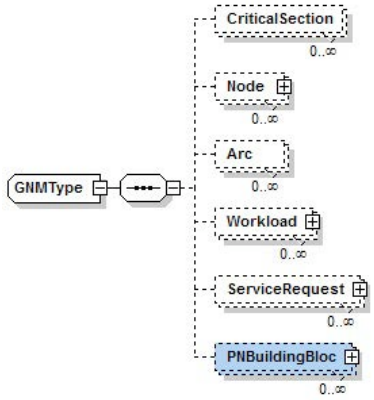

Figure 5: GPMIF schema with BBs.

```
  <graphics>
    <position x="285.0" y="180.0"/>
  </graphics>
  <name>
    <value>RAID-1</value>
    <graphics>
      <offset x="-5.0" y="35.0"/>
    </graphics>
  </name>
  <initialMarking>
    <value>0</value>
    <graphics>
      <offset x="0.0" y="0.0"/>
    </graphics>
  </initialMarking>
  <capacity>
    <value>0</value>
  </capacity>
 </place>
 ...
```

The full example and the schema is at `dmi.uib.es/~cllado/mifs/`.

5. EQUILIBRIUM SOLUTIONS OF GPMIF SPECIFICATIONS

The steady state probabilities of GPMIF models (when equilibrium exists) may be obtained by direct solution of the underlying Markov process by a standard numerical method, finding a product-form solution if such exists and simulation. We have implemented the second and third of these; the first could easily be done using the same specification as that provided to the simulator and software such as Dnamaca [15].

5.1 Product-forms by RCAT

The most general form of the RCAT theorem applies to pairwise synchronisations amongst any finite number of Markov processes [10]. In the present study a simpler version is sufficient, which is stated in the Appendix. Consider an isolated building block, which is open by definition. Unless there are population constraints at any of the BB's places – e.g. finite capacity – all the inputs to every place are outgoing from ev-

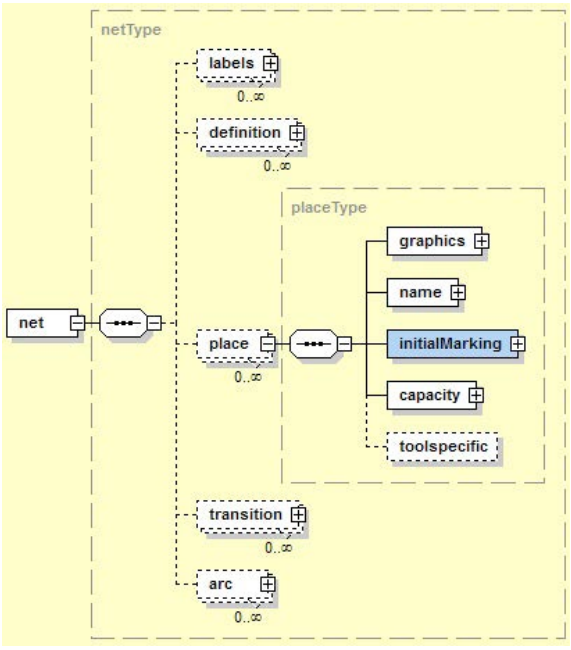

Figure 6: PNML schema included in the GPMIF.

ery state; this is because in building blocks they come from input transitions which are always enabled. Similarly, there is no restriction on the state from which a BB generates an output; hence every output causes a transition into every state of a building block. Thus RCAT can be applied, and a product-form therefore constructed in a network of BBs, provided the reversed rates of each output transition is the same at every one of its instances. This is not always the case, but the property does hold when the BB is a reversible Markov process. This is determined by the following result.

PROPOSITION 1. *Consider a BB-n, S with n places, and let $\mathcal{N} \subseteq 2^{\{1,\ldots,n\}} \setminus \emptyset$. Let $\rho_y = \lambda_y/\mu_y$ for $T_y, T'_y \in \mathcal{T}, y \in \mathcal{N}, |y| \geq 1$. If the following system of equations has a unique solution ρ_i, $(1 \leq i \leq N)$:*

$$\begin{cases} \rho_y = \prod_{i \in y} \rho_i & \forall y: \ T_y, T'_y \in \mathcal{T} \wedge |y| > 1 \\ \rho_i = \frac{\lambda_i}{\mu_i} & \forall i: \ T_i, T'_i \in \mathcal{T}, 1 \leq i \leq n \end{cases} \quad (1)$$

then the BB's balance equations – and hence equilibrium probabilities when they exist – have product-form solution:

$$\pi(m_1, \ldots, m_n) \propto \prod_{i=1}^{n} \rho_i^{m_i}. \quad (2)$$

PROOF. We show that the BB-n is reversible when at equilibrium. By the symmetry between the input and output transitions in the definition of a BB, there is certainly either no transition between any given pair of states or a transition in each direction. Let the equilibrium probabilities be $\pi(\mathbf{m}) \equiv \pi(m_1, \ldots, m_n)$. Then the detailed balance equations for transitions between states $\pi(\mathbf{m})$ and $\pi(\mathbf{m}')$, where $m'_k = m_k + 1$ for $k \in y$ and $m'_k = m_k$ for $k \notin y$ are:

$$\pi(\mathbf{m})\lambda_y = \pi(\mathbf{m}')\mu_y \quad \text{or} \quad \pi(\mathbf{m})\rho_y = \pi(\mathbf{m}')$$

This equation is satisfied for all $y \in \mathcal{N}$, $T_y \in \mathcal{T}$ and hence for all transition-pairs. The BB is therefore reversible with the equilibrium probabilities stated in the proposition. □

It is well known that M/M/1 queues satisfy the conditions of RCAT when departures are interpreted as active synchronising action and arrivals passive actions. Since the M/M/1 queue is reversible, the reversed rate of a departure is simply the arrival rate of its reversed transition, a constant. For the BB, as noted already, the first two conditions of RCAT are satisfied because of the nature of input and output transitions. Similarly the reversed rates are constant (assuming the arrival rates are state-independent) provided that the conditions of Proposition 1 are satisfied, in this case, $\rho_{12} = \rho_1\rho_2$ or $\lambda_1\lambda_2\mu_{12} = \mu_1\mu_2\lambda_{12}$.

5.1.1 *Illustrative exemplar*

The baby application with three queues connected between the outputs and inputs of a BB-2 can now be analysed for a product-form solution. In terms of BBs, it can be expressed as a synchronisation amongst three BB-1 nodes (the queues) and the BB-2 node, as shown in Figure 7. The outputs of BB-2 synchronise as active transitions with rates μ_1, μ_{12}, μ_2 (from left to right in the figure) with the passive inputs of the BB-1 nodes, which are assigned variable rates y_1, y_{12}, y_2. Similarly, the outputs from the BB-1 nodes synchronise actively at rates $\lambda_1, \lambda_{12}, \lambda_2$ with the passive inputs of BB-2, which are assigned variable rates x_1, x_{12}, x_2.

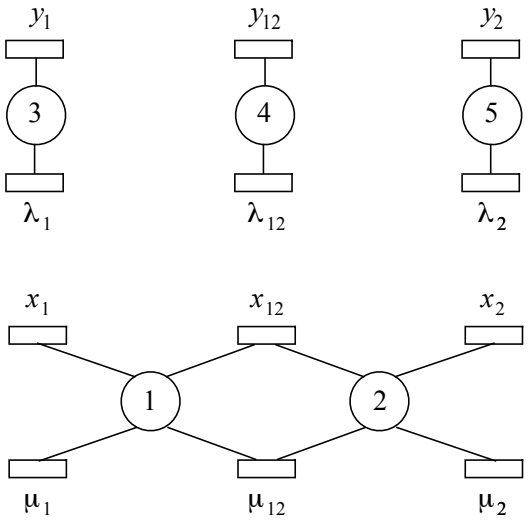

Figure 7: Composition of the hybrid network.

The rate equations for the three BB-2 outputs are now $y_1 = \overline{\mu_1} = x_1, y_{12} = \overline{\mu_{12}} = x_{12}, y_2 = \overline{\mu_2} = x_2$. For the three queues, we have similarly $x_1 = \overline{\lambda_1} = y_1, x_{12} = \overline{\lambda_{12}} = y_{12}, x_2 = \overline{\lambda_2} = y_2$, the same equations. We therefore have three degrees of freedom to choose x_1, x_2, x_3 say. However we have one constraint imposed by the BB-2 product-form condition,

$$x_{12} = \frac{\mu_{12} x_1 x_2}{\mu_1 \mu_2}$$

Therefore a product-form exists unconditionally since all the constraints can be satisfied. We have two degrees of freedom

in computing the unnormalised probabilities $\pi(n_1, n_2, n_3, n_4, n_5)$, where n_i is the number of tokens in place i in Figure 7; so let $x_1 = x_2 = 1$. Then we obtain

$$\pi(n_1, n_2, n_3, n_4, n_5) \propto \left(\frac{1}{\mu_1}\right)^{n_1} \left(\frac{1}{\mu_2}\right)^{n_2} \left(\frac{1}{\lambda_1}\right)^{n_3} \left(\frac{\mu_{12}}{\mu_1\mu_2\lambda_{12}}\right)^{n_4} \left(\frac{1}{\lambda_2}\right)^{n_5}$$

Whilst this is a nice product-form, normalisation still has to be done which requires enumeration of the state space, which is not entirely trivial. Because of the synchronisations in Petri net and the fact that it is closed, it is easy to verify that the total number of tokens in places 1, 3 and 4 is constant; symmetrically, the total number of tokens in places 2, 4 and 5 is constant. Therefore the state space comprises vectors $(n_1, n_2, n_3, n_4, n_5)$ such that $n_i \geq 0 (i = 1, \ldots, 5), n_1 + n_3 + n_4 = N_1, n_2 + n_4 + n_5 = N_2$ for given constants N_1, N_2. This turns out to give a state space of size $(N+1)(N+2)(2N+3)/6$ when $N_1 = N_2 = N$; te expression for $N_1 \neq N_2$ is rather messy.

Remark

The choice $x_1 = x_2 = 1$ can now be seen to be arbitrary. Leaving x_1, x_2 unspecified, we would have

$$\begin{aligned}\pi(n_1, n_2, n_3, n_4, n_5) &\propto \left(\frac{x_1}{\mu_1}\right)^{n_1} \left(\frac{x_2}{\mu_2}\right)^{n_2} \left(\frac{x_1}{\lambda_1}\right)^{n_3} \left(\frac{x_1 x_2 \mu_{12}}{\mu_1\mu_2\lambda_{12}}\right)^{n_4} \left(\frac{x_2}{\lambda_2}\right)^{n_5} \\ &= x_1^{n_1+n_2+n_4} x_2^{n_3+n_4+n_5} \times \left(\frac{1}{\mu_1}\right)^{n_1} \left(\frac{1}{\mu_2}\right)^{n_2} \left(\frac{1}{\lambda_1}\right)^{n_3} \left(\frac{\mu_{12}}{\mu_1\mu_2\lambda_{12}}\right)^{n_4} \left(\frac{1}{\lambda_2}\right)^{n_5} \\ &\propto \left(\frac{1}{\mu_1}\right)^{n_1} \left(\frac{1}{\mu_2}\right)^{n_2} \left(\frac{1}{\lambda_1}\right)^{n_3} \left(\frac{\mu_{12}}{\mu_1\mu_2\lambda_{12}}\right)^{n_4} \left(\frac{1}{\lambda_2}\right)^{n_5}\end{aligned}$$

since the powers of x_1 and x_2 are the constants N_1 and N_2 respectively.

However, it should be noted that this simple network is analogous to three parallel cyclic pairs of queues, with constraints imposed on the middle queue only. This is why there are the two degrees of freedom. A more general topology, in either a closed or an open network, would result in no degrees of freedom after normalisation so that the constraints arising from any BB-n nodes ($n > 1$) would restrict the domain of parameters allowing a product-form solution. Our case study in section 6.1 is an instance of this.

More simply, consider the cyclic network of Figure 4. The can be described similarly to the composition of Figure 7, except the passive actions y_1, y_{12}, y_2 must each be replaced by three passive actions corresponding to input from the three output transitions of the BB-2 node; denote these by y_{1a}, y_{1b}, y_{1c} in place of y_1 and similarly for y_{12} and y_2. The rate equations of RCAT now become:

$$y_{1a} = \overline{\mu_1}/3 = x_1/3, y_{1b} = x_{12}/3, y_{1c} = x_2/3$$

and $x_1 = \overline{\lambda_1} = y_1$, where $y_1 = y_{1a} + y_{1b} + y_{1c}$, $x_{12} = y_{12}$ and $x_2 = y_2$. Thus, $(x_1 + x_{12} + x_2)/3 = x_1 = x_{12} = x_2$ and so the condition for product-form in Proposition 1 becomes

$$x_1 = x_{12} = x_2 = y_1 = y_{12} = y_2 = \mu_1\mu_2/\mu_{12}$$

The condition can therefore be satisfied, giving the unconditional product-form

$$\pi(n_1, n_2, n_3, n_4, n_5) \propto \left(\frac{\mu_2}{\mu_{12}}\right)^{n_1} \left(\frac{\mu_1}{\mu_{12}}\right)^{n_2} \left(\frac{\mu_1\mu_2}{\lambda_1\mu_{12}}\right)^{n_3} \left(\frac{\mu_1\mu_2}{\mu_{12}\lambda_{12}}\right)^{n_4} \left(\frac{\mu_1\mu_2}{\lambda_2\mu_{12}}\right)^{n_5}$$

5.1.2 Open networks

An open network would be specified in a similar way except that certain queues would have external arrival streams and external departure streams probabilistically chosen after a service completion; these streams have fixed rates or selection probabilities rspectively and do not synchronise with other nodes. Similarly, BBs may have additional input and/or output transitions that do not synchronise. Suppose we add an additional external arrival stream into the leftmost queue (labelled 3) with rate γ_1 and external departures with probability 0.5 from the rightmost queue (labelled 5). The rate equations now become

$$x_1 = \gamma_1 + y_1, x_{12} = y_{12}, x_2 = y_2/2$$

$$y_{ua} = x_1/3, y_{ub} = x_{12}/3, y_{uc} = x_2/3 \quad \text{(u=1,12,2)}$$

Thus, $x_{12} = (x_1 + x_{12} + x_2)/3, x_2 = (x_1 + x_{12} + x_2)/6, x_1 = \gamma_1 + (x_1 + x_{12} + x_2)/3$. These equations have the unique solution $x_1 = 3\gamma_1, x_{12} = 2\gamma_1, x_2 = \gamma_1$. There is therefore a product-form solution if and only if $2\mu_1\mu_2 = 3\gamma_1\mu_{12}$, whereupon it is

$$\pi(n_1, n_2, n_3, n_4, n_5) \propto \left(\frac{3\gamma_1}{\mu_1}\right)^{n_1} \left(\frac{\gamma_1}{\mu_2}\right)^{n_2} \left(\frac{3\gamma_1}{\lambda_1}\right)^{n_3} \left(\frac{2\gamma_1}{\lambda_{12}}\right)^{n_4} \left(\frac{\gamma_1}{\lambda_2}\right)^{n_5}$$

5.2 Translation into a SPN

Performance Model interoperability between Queueing Networks and Petri nets is desirable since it can be very useful to compare performance results coming from tools that use different formalisms and the distinct benefits of the tools can be shared and combined. In our case, we can take advantage of the efficiency of product-form solutions on the one hand and be able to solve general models, without the constraints that hamper product-forms, on the other. In [5] a tool that allows for the transformation of a QN specified using PMIF into a Stochastic Petri Net (SPN) is presented. The resulting SPN can be read and solved by PIPE2 (Platform Independent Petri net Editor 2) [4] and TimeNet [22]. The QN → PN tool uses the ATL transformation language to translate from the PMIF schema to a SPN schema (*eDSPN.xsd*) used by TimeNet and that can also be inported/exported to/from PIPE2 . This way, we can use both tools dependent on which one is more convenient. We enhance this transformation tool such that it can also transform PMIF models with BBs (we call it eQN->PN, for "enhanced QN->PN tool"). Since the BBs are actually SPNs anyway, the transformation enhancement is mainly syntactic. We can then use the eQN->PN tool to tranform *any* PMIF model with BBs (product-form or not, open or closed) into a SPN model that can be solved by PIPE. If the PN is bounded, we can use a GSPN analysis solver such as PIPE's one to obtain equilibrium probabilities when they exist [4].

6. MODELLING RAID SYSTEMS

Redundant Arrays of Inexpensive Disks (RAIDs) have been used for cost-efficient storage, increased performance through parallelism and fault tolerance for many years [3]. A RAID

subsystem is problematic to represent in a queueing model since it involves a fork-join operation, whereby an arriving task, representing an access request, *forks* into a number of subtasks that each go to a different disk. This is because data is "striped", i.e. divided up into a number of segments that are allocated to different disks in the array, and/or mirrored, i.e. copied to another disk. The subtasks are run asynchronously, perhaps queueing with subtasks from other accesses, and then recombined after all have been served, i.e. are *joined*. Stripes can be of any size (number of segments, or disks used) up to the number of disks in the array, n say. Large accesses will use a number of full stripes of n segments and a partial stripe of size less than n. Smaller requests will just comprise a partial stripe. We assume the partial stripes use a sequence of adjacent disks, with wraparound, starting at any disk with equal probability. Thus, in a BB-n, we need to specify one workload for each possible combination of disks used for each stripe size. There is only one for a full stripe – the compete set of n disks. For partial stripes of size $k < n$, there are n combinations of disks that can be selected, corresponding to the choice of disk for the first segment in the stripe. Hence there are $1 + n(n-1)$ workloads in the PMIF, corresponding to each of these combinations.

A RAID system of n disks is modelled here by a BB-n. This can represent the forking of arrivals into up to n subtasks, which pass to places at which they are served with processor-sharing discipline as in standard Petri nets. Moreover, the building block can also represent the corresponding combining operations. However, it does not faithfully model joins because the subtasks output in parallel are selected randomly from each input place and do not in general correspond to the same task that forked previously. Nevertheless, it is a good approximation at low utilisations – in fact exact in the limit that the occupancy of the places never exceeds 1. Moreover, at equilibrium, on average the number of forks over a long period will be equal to the number of corresponding joins. Obviously a further limitation is that individual disk service times are assumed to be exponential random variables, but this is common to the other servers in a larger system-model; and indeed to many analytical models prevalent in performance engineering. We anticipate reasonable accuracy in the prediction of system measures such as mean place occupancy levels, device utilisations and throughput, but probably poor on user-oriented measures like response time variance and probability distribution.

6.1 Case study

We constructed a model of a RAID within a typical, interactive, multi-access data storage system. As discussed above, the RAID is modelled by a BB-n node and the other service centres are modelled by conventional queues. The model is depicted in Figure 8 and implemented first, for simplicity of explanation, with a BB-2 node. The method is easy to mechanise and it is straightforward to introduce the additional workloads required for $n > 2$.

We have assumed that there are two other dedicated disks, A and B, a CPU and a "think" node, with infinite server discipline, that represents user interaction in a multi-access system (historically this would be a "terminal system"). We solve this model, when equilibrium exists, first by product-form solution and then by translation into a standard Petri net by PIPE; see Figure 9. Notice that the Petri net is unbounded, with infinite state space. In the general case, with a BB-n representation of the RAID, when the total token-population is N, suppose both the input to T_{12} and the output from T'_{12} are suspended. The remaining operational transitions are now equivalent to an irreducible queueing network and so any configuration of the tokens amongst the six places is possible, subject to the total population of N. Moreover, if the input to T_{12} operates once, the population will go up to $N+1$ and repeating the preceding argument shows that any configuration is possible with total population $N+1$. Conversely, if T'_{12} fires once, the population reduces by one. Hence we see inductively that any state $(n_1, \ldots, n_6)$, $n_i \geq 0, 1 \leq i \leq 6$ is reachable. This is a complication for simulation or direct solution, which must truncates the state space appropriately, but a great simplification for any product-form solution which does not need to find a normalising constant (or, rather, can determine it easily as the product of the normalising constants of the individual nodes).

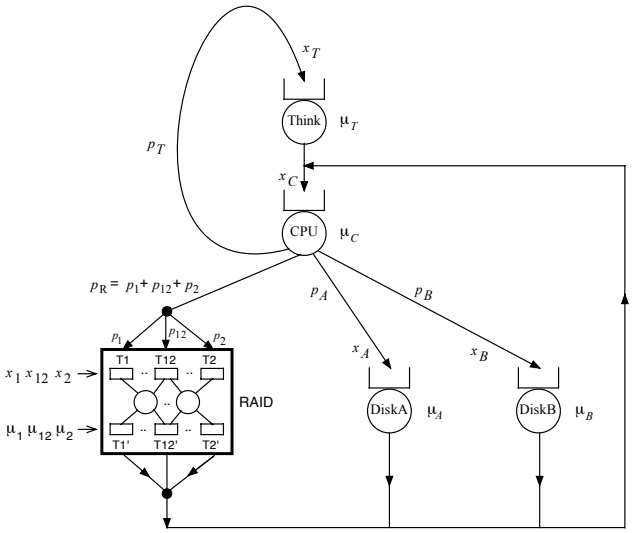

Figure 8: RAID network model.

6.2 Product-form solution

In this model, every active action represents a service completion at a queue or the firing of an output transition in the BB node. In each case, the reversed rate is the arrival rate of the individual queue or the rate of the corresponding input transition. We can therefore write down the rate equations of RCAT as follows, for the case of a BB-2 RAID-node. First note that the total arrival rate to the CPU queue, x_C in Figure 8, is the sum of the passive arrival rates from the RAID BB and the passive arrival rates from DiskA, DiskB and the Think queues.

$$\begin{aligned} x_T &= p_T x_C \\ x_1 &= p_1 x_C \\ x_{12} &= p_{12} x_C \\ x_2 &= p_2 x_C \\ x_A &= p_A x_C \\ x_B &= p_B x_C \\ x_C &= x_T + x_1 + x_{12} + x_2 + x_A + x_B \end{aligned}$$

where the terms $p_\bullet$ are the routing probabilities indicated in Figure 8. This is a homogeneous set of linear equations with a unique solution up to a constant multiplicative factor $\mathbf{x} = (p_T, p_1, p_{12}, p_2, p_A, p_B, 1)x_C$. There is therefore one degree of freedom, since the normalising constant is the product of the individual node-normalising constants. The degree of freedom is used up by the single constraint of the BB-2 node:

$$x_1 x_2 \mu_{12} = \mu_1 \mu_2 x_{12}$$

which implies that

$$x_C = \frac{\mu_1 \mu_2 p_{12}}{\mu_{12} p_1 p_2}$$

The unconditional product-form (when there is equilibrium) is therefore:

$$\pi(n_1, \ldots, n_6)) = e^{-z_1} \prod_{i=2}^{6} (1 - z_i) \frac{1}{n_1!} \prod_{i=1}^{6} z_i^{n_i} \qquad (3)$$

where we rename the variables as $z_1 = p_T x_C/\mu_T$, $z_2 = p_1 x_C/\mu_1$, $z_3 = p_2 x_C/\mu_2$, $z_4 = p_A x_C/\mu_A$, $z_5 = p_B x_C/\mu_B$, $z_6 = x_C/\mu_C$. Finally, the condition for equilibrium to exist is $z_i < 1$ for $2 \leq i \leq 6$, i.e.

$$x_C < \min(\mu_1/p_1, \mu_2/p_2, \mu_A/p_A, \mu_B/p_B, \mu_C)$$

6.2.1 Numerical parameterisation

The parameter values chosen for the model depicted in Figure 8 are as shown in the specification in section 4.1 for the BB-2 node. The complete parameterisation is, for the service rates: $\mu_T = 1/60, \mu_1 = 5, \mu_{12} = 12, \mu_2 = 5, \mu_A = 50, \mu_B = 20, \mu_C = 100$; and for the routing probabilities $p_T = 1/16, p_1 = 1/16, p_{12} = 1/8, p_2 = 1/16, p_A = 7/16, p_B = 1/4$. From these we calculate the mean queue lengths at all the nodes, $m_i, i = 1, \ldots, 6$, the throughput τ and mean response time R for a user. These follow directly from the equilibrium probabilities 3 and Little's result as:

$$m_1 = z_1; m_i = z_i/(1 - z_i), \quad i = 2, \ldots, 6$$

and

$$\tau = z_1 \mu_T = x_C p_T; R = (m_1 + \ldots + m_6)/\tau$$

Numerical results for the performance predicted by this model are reported in section 6.4, where we also look at the changes we get by increasing the rate μ_{12} of the output transition T'_{12} (representing the access time for stripes of size two) from 12 to 16.

6.2.2 Constraints for product-forms with BB-n

Similar RAID systems modelled with BB-n nodes are no more difficult to analyse but there are more RAID inputs – actually $1 + n(n-1)$ of them – to define, as discussed above. Let us assume that the output rates are fixed and try to choose the input probabilities such that the conditions for product-form are satisfied. Since there are $1 + n(n-1)$ inputs, there are $n(n-1)$ independent probabilities and 1 degree of freedom because the network is closed with homogeneous equations for the rates $x_\bullet$. The number of constraints in Proposition 1 is the number of inputs less the number of places, i.e. $1 + n(n-1) - n = (n-1)^2$ in the RAID model, the number of inputs that fork. Subtracting the one degree of freedom, we have $n(n-2)$ further constraints to satisfy, which is n less than the number of independent input probabilities to assign, $n(n-1)$. Therefore, a product-form can be guaranteed by picking the input probabilities appropriately. This observation allows file allocations to be organised so as to achieve product-forms, facilitating simple quantitative analysis.

In particular, since the number of free input probabilities is $n(n-1) - n(n-2) = n$, which is the number of inputs that do not fork, one strategy is to choose the non-forking input probabilities in any way desired, for example proportional to the rates of the corresponding disk drives. The probabilities for the forking tasks are then defined uniquely if the network is to have a product-form. Thus, for $n = 2$ above, there are no further constraints to satisfy and, as we found, the product-form is unconditional; we can choose the two non-forking input probabilities, whereupon the remaining (forking) one is determined. For $n = 3$, the BB-3 has 7 inputs, 4 constraints, 1 degree of freedom and 6 independent input probabilities. Hence a product-form can be found with three of the input probabilities assigned arbitrarily; i.e. we can again choose the non-forking probabilities, after which the other four are determined.

6.3 Direct solution of a Petri net model

The GPMIF xml specification corresponding to the model of Figure 8 (partially written in section 4) will be transformed by our eQN->PN tool in another xml file specifying a Petri net that can be read by PIPE. This net is shown in Figure 9. It is an standard SPN where all timed transitions have single server semantics but the *THINKTime* transition that needs an infinite server semantics in order to model the *Think Device*.

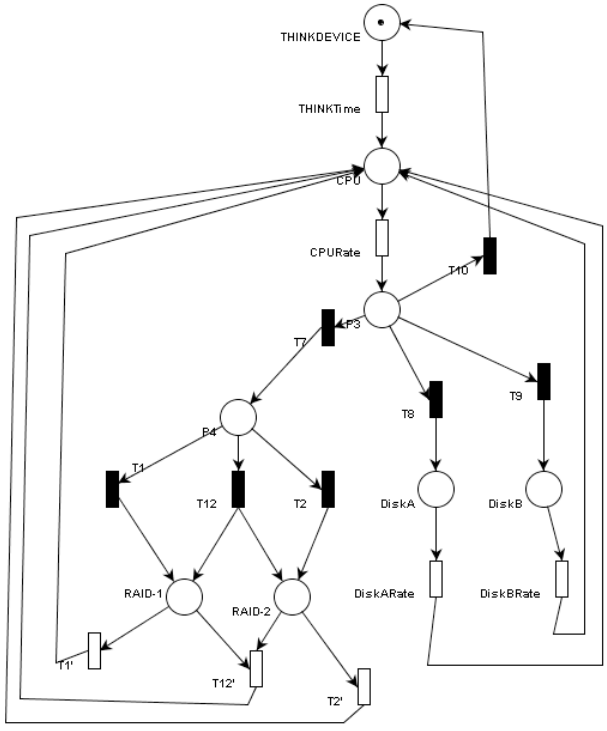

Figure 9: Petri net model produced by PIPE.

6.4 Numerical results

For the model defined in section 6.2.1, we obtained the mean queue lengths, or numbers of tokens at places shown in Table 1 as well as an average of 5 tokens in RAID-1 and RAID-2; the non-think node utilisations were 0.833 (RAID-1), 0.833 (RAID-2), 0.583 (DiskA), 0.833 (DiskB), 0.667 (CPU); the throughput was 4.167; the mean network population was 268.4; and the mean response time was 4.416 seconds. As it can be seen from the table, these results were confirmed very closely by a discrete event simulation of the PIPE-generated Petri net.

When we changed the output rate of the full stripes (transition T'_{12}) from 12 to 16, the corresponding results were, were again confirmed closely by the simulation as also shown in Table 1. For the mean node occupancies of RAID-1 and RAID-2 we got 1.667 and the non-think node utilisations were 0.625 (RAID-1), 0.625 (RAID-2), 0.438 (DiskA), 0.625 (DiskB), 0.5 (CPU); the throughput became 3.125; the mean network population was 194.3; and the mean response time was 2.169 seconds.

Server	Method	T'_{12}=12	T'_{12}=16
Thinking	Product-form	250.0	187.5
	Simulation	249.5	187.3
DiskA	Product-form	1.4	0.778
	Simulation	1.41	0.773
DiskB	Product-form	5.0	1.667
	Simulation	5.01	1.660
CPU	Product-form	2.0	1.0
	Simulation	2.0	0.995

Table 1: Numerical results comparison

6.5 Arbitrary sized RAID arrays

As we discussed in section 6.2, in an array of $d \geq 2$ disks, a product-form is unconditional if there are only $d+1$ different input combinations, since this implies only d independent input probabilities. In the special case that a RAID IO request is either to just one (any) single disk or to a full stripe across all disks, we require only the transitions $T_1, \ldots, T_d$ and $T_{1,2,\ldots,d}$ in the building block BB-d. With obvious extension of notation, this gives the constraint $x_1 \ldots x_d \mu_{1\ldots d} = \mu_1 \ldots \mu_d x_{1\ldots d}$, leading to

$$x_C^{d-1} = \frac{p_{1\ldots d}\mu_1 \ldots \mu_d}{\mu_{1\ldots d} p_1 \ldots p_d}$$

which gives (in the notation of section 6.2) the product-form solution

$$\pi(n_1, \ldots, n_6)) = e^{-z_1} \prod_{i=2}^{d+4}(1 - z_i)\frac{1}{n_1!}\prod_{i=1}^{d+4} z_i^{n_i}$$

with no further conditions.

Any additional combinations of inputs in the BB-d, such as would be required to model partial stripes, for example, would lead to conditions on the rates or the input selection probabilities for the product-form to be valid.

7. CONCLUSION

By facilitating sharing of software model specifications, portability and ease of use, PMIFs are enhancing the performance engineering process and making it accessible to the non-specialist. We have made a significant extension to the GPMIF schema so as to allow fork-join sub-models and a subset of Petri nets to be specified and checked automatically for efficient, product-form solutions. The compositional approach, using queues and Petri net "building blocks" is naturally hierarchical and conducive to application of RCAT, which is what provides product-forms when they exist. Otherwise, solutions for general GPMIF model specifications can be obtained by direct solution of the Petri net form of the model, using either numerical solution of its Markov chain or simulation. We have focused on the latter, using the simulation component of Dnamaca, which can also solve for the equilibrium probabilities (when they exist) of the Markov chain when the state space is truncated suitably. The Petri net derived for the RAID system model is unbounded and so requires truncation, so direct analytic solution has been left for future work. Nevertheless, our product-form solutions show that, under the given approximating assumptions, RAID-type fork and join operations can be incorporated efficiently into standard queueing network models.

In the immediate future, we plan to incorporate general BB-n building blocks for any integer n, including the mechanical checking of the conditions for product-form by Proposition 1. This will make our RAID system models more realistic and suitable for testing against data monitored on real systems. Longer term we intend to apply our approach to more realistic case studies, to find more general building blocks for fork-join that do not require exponential service times for the sub-tasks (e.g. relax these to Erlang or Coxian) and provide a beta-version implementation.

Acknowledgments

This work is partially funded by the TIN2010-16345 EIGER project of the *Ministerio de Educacion y Ciencia*, Spain.

8. REFERENCES

[1] Platform Independent Petri net Editor 2. `http://pipe2.sourceforge.net/`.

[2] Balsamo, S., Harrison, P., and Marin, A. Systematic construction of product-form stochastic petri-nets. Tech. rep., Universita CaÕ Foscari Venezia, August 2009.

[3] board, T. R. A. *The RAIDBOOK : A source Book for RAID Technology*. Lino Lakes MN Publisher, June 1993.

[4] Bonet, P., Llado, C., Puijaner, R., and Knottenbelt., W. PIPE v2.5: A Petri net tool for performance modelling. In *Proc. 23rd Latin American Conference on Informatics* (San Jose, Costa Rica, 9-12 October 2007).

[5] Bonet, P., Llado, C., Smith, C. U., and Puijaner, R. Pmif->pn. In *Submitted for publication* (2011).

[6] Boucherie, R. J. A characterisation of independence for competing Markov chains with applications to stochastic Petri nets. *IEEE Trans. on Software Eng. 20*, 7 (1994), 536–544.

[7] Coleman, J. L., Henderson, W., and Taylor, P. G. Product form equilibrium distributions and a convolution algorithm for Stochastic Petri nets. *Perf.*

Eval. 26, 3 (1996), 159–180.

[8] Gelenbe, E. Random neural networks with positive and negative signals and product form solution. *Neural Computation 1*, 4 (1989), 502–510.

[9] Harrison, P. Turning back time in Markovian process algebra. *Theoretical Computer Science 290*, 3 (2003), 1947–1986.

[10] Harrison, P., and Lee, T. Separable equilibrium state probabilities via time reversal in Markovian process algebra. *Theoretical Computer Science* (2005).

[11] Harrison, P., Lladó, C., and Puigjaner, R. A general performance model interchange format. In *Proc. of the First International Conference on Performance Evaluation Methodologies and Tools (Valuetools)* (2006).

[12] Harrison, P., Lladó, C., and Puigjaner, R. A unified approach to modelling the performance of concurrent systems. *Special Issue on "Advances in System Performance Modelling, Analysis and Enhancement", Simulation Modelling Practice and Theory 17*, 9 (October 2009), 1445–1456.

[13] Henderson, W., Lucic, D., and Taylor, P. G. A net level performance analysis of Stochastic Petri Nets. *J. Austral. Math. Soc. Ser. B 31* (1989), 176–187.

[14] Iacono, M., and Gribaudo, M. Element based semantics in multi formalism performance models. In *Proc. of the 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (2010), IEEE Computer Society.

[15] Knottenbelt, W., Harrison, P., and Kritzinger, P. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation 39* (2000), 127–148.

[16] Mairesse, J., and Nguyen, H.-T. Deficiency Zero Petri Nets and Product Form. In *Proc. of the 30th Int. Conf. on App. and Theory of Petri Nets* (Paris, France, 2009), PETRI NETS '09, Springer-Verlag, pp. 103–122.

[17] Sanders, W. H., Courtney, T., Deavours, D., Daly, D., Derisavi, S., and Lam, V. Multi-formalism and multi-solution-method modeling frameworks: The mobius approach. `https://www.perform.csl.illinois.edu/Papers/USAN_papers/03SAN01.pdf`.

[18] Smith, C., and Llado, C. Performance model interchangeformat (pmif 2.0): Xml definition and implementation. In *Proc. of the First International Conference on the Quantitative Evaluation of Systems* (September 2004), pp. 38–47.

[19] Smith, C. U., Lladó, C. M., and Puigjaner, R. Performance Model Interchange Format (PMIF 2): A comprehensive approach to queueing network model interoperability. *Performance Evaluation 67*, 7 (2010), 548 – 568.

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

[21] Vittorini, V., Iacono, M., Mazzocca, N., and Franceschinis, G. The osmosys approach to multi-formalism modeling of systems. *Software and Systems Modeling 3* (2004), 68–81.

10.1007/s10270-003-0039-5.

[22] Zimmermann, A., and Knoke, M. TimeNET 4.0. a Software Tool for the Performability Evaluation with Stochastic and Coloured Petri Nets. User Manual. Tech. rep., Technische Universität Berlin. Real-Time Systems and Robotics Group, August 2007.

APPENDIX

A. MULTI-AGENT REVERSED COMPOUND AGENT THEOREM (RCAT)

Expressed using an extension of PEPA, consider a multiple component, pairwise synchronisation amongst n component-processes $P_1, \ldots, P_n$: $\underset{L}{\overset{n}{\bowtie}}_{k=1} P_k$ $(n \geq 2)$, where $L = \bigcup_{k=1}^{n} L_k$ and L_k is the set of synchronising action types that occur in P_k. Every action in each of the n components synchronises with at most one other, such that one instance of the action type is active and the other is passive. A special case of the Multiple Agent Reversed Compound Agent Theorem (MARCAT)[3] defines a product-form solution for the steady state probabilities of the synchronised Markov process under certain conditions, when equilibrium exists.

Theorem 1. *(MARCAT)*
Suppose that the cooperation $\underset{L}{\overset{n}{\bowtie}}_{k=1} P_k$ *of components* P_k*, denoting stationary CTMCs, has a state transition graph with an irreducible subgraph* G *and that the cooperation set* L *is finite. Let* $R_k = P_k\{\top_a \leftarrow x_a \mid a \in \mathcal{P}_k\}$ *for* $k = 1, \ldots, n$*. Given that*

1. *For* $k = 1, 2, \ldots, n$*, every passive action type in* P_k *is always enabled, i.e.* outgoing *in all states of* P_k*;*
2. *For* $k = 1, 2, \ldots, n$*, every active action type in* P_k *is* incoming *in all states of* P_k*;*
3. *Every instance of a reversed action, type* $\overline{a}$*, of an* active *action type* $a \in \mathcal{A}_k$ *has the same rate* $\overline{r_a}$ *in* $\overline{R_k}$ *(*$1 \leq k \leq n$*), and* $\{x_a\}$ *satisfy the* rate equations

$$\{x_a = \overline{r_a} \mid a \in \mathcal{A}_k, 1 \leq k \leq n\}$$

then the synchronisation has product-form solution $\pi(\underline{i}) \propto \prod_{k=1}^{n} \pi_k(i_k)$ *for the equilibrium probability of state* $\underline{i} = (i_1, \ldots, i_n)$*, where* $\pi_k(i_k)$ *is proportional to the equilibrium probability of state* i_k *in* R_k*.*

The proof verifies Kolmogorov's criteria for reversed processes and is given in full in [10].

[3]The most general form, which we do not need here, is given in [10]