Statistical Model Checking QoS properties of Systems with SBIP *

Saddek Bensalem[1], Marius Bozga[1], Benoit Delahaye[2],
Cyrille Jegourel[3], Axel Legay[3], and Ayoub Nouri[1]

[1] UJF-Grenoble 1 / CNRS VERIMAG UMR 5104, Grenoble, F-38041, France
[2] Aalborg University, Denmark
[3] INRIA/IRISA, Rennes, France

Abstract. BIP is a component-based framework supporting rigorous design of embedded systems. This paper presents SBIP, an extension of BIP that relies on a new stochastic semantics that enables verification of large-size systems by using Statistical Model Checking. The approach is illustrated on several industrial case studies.

1 Introduction

Expressive modeling formalism with sound semantical basis and efficient analysis techniques are essential for successful model-based development of embedded systems. While expressivity is needed for mastering heterogeneity and complexity, sound and rigorous models are mandatory to establish and reason meaningfully about system correctness and performance at design time.

The BIP (Behaviour-Interaction-Priority) [3] formalism is an example of a highly expressive, component-based framework with rigorous semantical basis. BIP allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Such components are transition systems enriched with variables. Transitions are used to move from a source to a destination location. Each time a transition is taken, component variables may be assigned new values, possibly computed by C functions. Atomic components are composed by layered application of interactions and priorities. Interactions express synchronization constraints between actions of the composed components while priorities are used both to select amongst possible interactions and to steer system evolution so as to meet performance requirements e.g. to express scheduling policies. BIP is supported by an extensible toolset which includes tools for checking correctness, for model transformations and for code generation. Correctness can be either formally proven using invariants and abstractions, or tested using simulation. For the latter case, simulation is driven by a specific middleware, the BIP engine, which allows to generate and

* Research supported by the European Community's Seventh Framework Programme [FP7] under grant agreements no 248776 (PRO3D), no 288917 (DALI), no 287716 (DANSE), no 257414 (ASCENS), the ARTEMIS JU grant agreement 2009-1-100208 (ACROSS), and Regional CREATIVE project ESTASE.

explore execution traces corresponding to BIP models. Model transformations allow to realize static optimizations as well as special transformations towards distributed implementation of models. Finally, code generation targets both simulation and implementation models, for different platforms and operating systems support (e.g., distributed, multi-threaded, real-time, etc.). The tool has been applied to a wide range of academic case studies as well as to more serious industrial applications [5].

BIP is currently equiped with a series of runtime verification [8] and simulation engines. While those facilities allow us to reason on a given execution, they cannot be used to assess the overall correctness of the entire system. This paper presents SBIP, a stochastic extension of the BIP formalism and toolset. Adding stochastic aspects permits to model uncertainty in the design e.g., by including faults or execution platform assumptions. Moreover, it allows to combine the simulation engine of BIP with statistical inference algorithms in order to reason on properties in a quantitative manner. Stochastic BIP relies on two key features. The first is a stochastic extension of the syntax and the semantics of the BIP formalism. This extension allows us to specify stochastic aspects of individual components and to produce execution traces of the designed system in a random manner. The second feature is a Statistical Model Checking (SMC) [25, 28, 16, 23, 4, 30, 29, 17] engine (SBIP) that, given a randomly sampled finite set of executions/simulations of the stochastic system, can decide with some confidence whether the system satisfies a given property. The decision is taken through either a Monte Carlo (that estimates the probability) [9], or an hypothesis testing algorithm [28, 25] (that compares the probability to a threshold). Due to SMC restrictions, these properties must be evaluated on bounded executions. Here, we restrict ourselves to Bounded Linear Temporal Logic (BLTL). As it relies on sampling executions of a unique distribution, SMC can only be applied to pure stochastic systems i.e., systems without non-determinism. The problem is that most of component-based design approaches exhibit non-determinism due to interleaving semantics, usually adopted for parallel execution of components and their interactions. SBIP allows to specify systems with both non-deterministic and stochastic aspects. However, the semantics of such systems will be purely stochastic, as explained hereafter. Syntactically, we add stochastic behaviour to atomic components in BIP by randomizing individual transitions. Indeed, it suffices to randomize the assignments of variables, which can be practically done in the C functions used on transition. Hence, from the user point of view, dealing with SBIP is as easy as dealing with BIP.

Our approach is illustrated on several case studies that cannot be handled with existing model checkers for stochastic systems [20, 15]. The presentation restricts to the analysis of a clock synchronization protocol [1] and an MPEG decoder. Other examples can be found in [2].

Structure of the paper. Section 2 presents the BIP framework. The stochastic extension for BIP and its associated semantics are introduced in Section 3. Section 4 describe the Probabilistic Bounded Linear Time Logic, the statistical

model checking procedure as well as the implementation of our extension in BIP. Finally, Sections 5 and 6 present experiments and conclusion, respectively.

2 Background on BIP

The BIP framework, presented in [3], supports a methodology for building systems from *atomic components*. It uses *connectors*, to specify possible interactions between components, and *priorities*, to select amongst possible interactions.

Atomic components are finite-state automata that are extended with variables and ports. Variables are used to store local data. Ports are action names, and may be associated with variables. They are used for interaction with other components. States denote control locations at which the components await for interaction. A transition is a step, labeled by a port, from a control location to another. It has associated a guard and an action that are, respectively, a Boolean condition and a computation defined on local variables. In BIP, data and their related computation are written in C. Formally:

Definition 1 (Atomic Component in BIP). *An atomic component is a transition system extended with data* $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$*, where:*

- (L, P, T) *is a transition system, with* $L = \{l_1, l_2, \ldots, l_k\}$ *a set of control locations,* P *a set of ports, and* $T \subseteq L \times P \times L$ *a set of transitions,*
- $X = \{x_1, \ldots, x_n\}$ *is a set of variables over domains* $\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ *and for each* $\tau \in T$ *respectively,* $g_\tau(X)$ *is a guard, a predicate on* X*, and* $X' = f_\tau(X)$ *is a deterministic update relation, a predicate defining* X' *(next) from* X *(current) state variables.*

For a given valuation of variables, a transition can be executed if the guard evaluates to true and some *interaction* involving the port is enabled. The execution is an atomic sequence of two microsteps: 1) execution of the interaction involving the port, which is a synchronization between several components, with possible exchange of data, followed by 2) execution of internal computation associated with the transition. Formally:

Definition 2 (Semantics of atomic component). *The semantics of* $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$ *is a transition system* (Q, P, T_0) *such that*

- $Q = L \times \mathbf{X}$ *where* $\mathbf{X}$ *denotes the set of valuations* v_X *of variables in* X*.*
- T_0 *is the set including transitions of the form* $((l, v_X), p, (l', v'_X))$ *such that* $g_\tau(v_X) \wedge v'_X = f_\tau(v_X)$ *for some* $\tau = (l, p, l') \in T$*. As usual, if* $((l, v_X), p, (l', v'_X)) \in T_0$*, we write* $(l, v_X) \xrightarrow{p} (l', v'_X)$*.*

Composite components are defined by assembling sub-components (atomic or composite) using *connectors*. Connectors relate ports from different subcomponents. They represent sets of interactions, that are, non-empty sets of ports that have to be jointly executed. For every such interaction, the connector provides the guard and the data transfer, that are, respectively, an enabling

condition and an exchange of data across the ports involved in the interaction. Formally:

For a model built from a set of component $B_1, B_2, \ldots, B_n$, where $B_i = (L_i, P_i, T_i, X_i, \{g_\tau\}_{\tau \in T_i}, \{f_\tau\}_{\tau \in T_i})$ we assume that their respective sets of ports and variables are pairwise disjoint, i.e. for any two $i \neq j$ in $\{1 \ldots n\}$, we require that $P_i \cap P_j = \emptyset$ and $X_i \cap X_j = \emptyset$. Thus, we define the set $P = \bigcup_{i=1}^{n} P_i$ of all ports in the model as well as the set $X = \bigcup_{i=1}^{n} X_i$ of all variables.

Definition 3 (Interaction). *An interaction a is a triple (P_a, G_a, F_a) where $P_a \subseteq P$ is a set of ports, G_a is a guard, and F_a is a data transfer function. We restrict P_a so that it contains at most one port of each component, therefore we denote $P_a = \{p_i\}_{i \in I}$ with $p_i \in P_i$ and $I \subseteq \{1 \ldots n\}$. G_a and F_a are defined on the variables available on the interacting ports $\bigcup_{p \in a} X_p$.*

Given a set of interactions γ, the composition of the components following γ is the component $B = \gamma(B_1, \ldots, B_n) = (L, \gamma, \mathcal{T}, X, \{g_\tau\}_{\tau \in \mathcal{T}}, \{f_\tau\}_{\tau \in \mathcal{T}})$, where $(L, \gamma, \mathcal{T})$ is the transition system such that $L = L_1 \times \ldots \times L_n$ and $\mathcal{T} \subseteq L \times \gamma \times L$ contains transitions of the form $\tau = ((l_1, \ldots, l_n), a, (l'_1, \ldots, l'_n))$ obtained by synchronization of sets of transitions $\{\tau_i = (l_i, p_i, l'_i) \in T_i\}_{i \in I}$ such that $\{p_i\}_{i \in I} = a \in \gamma$ and $l'_j = l_j$ if $j \notin I$. The resulting set of variables is $X = \cup_{1 \leq i \leq n} X_i$, and for a transition τ resulting from the synchronization of a set of transitions $\{\tau_i\}_{i \in I}$, the associated guard (resp. update relation) is the conjunction of the individual guards (resp. update relations) involved in the transition.

Finally, *priorities* provide a means to coordinate the execution of interactions within a BIP system. They are used to specify scheduling or similar arbitration policies between simultaneously enabled interactions. More concretely, priorities are rules, each consisting of an ordered pair of interactions associated with a condition. When the condition holds and both interactions of the corresponding pair are enabled, only maximal one can be executed. Non-determinism appears when several interactions are enabled. In the following, when we introduce probabilistic variables, we will thus have to make sure that non-determinism is resolved in order to produce a purely stochastic semantics.

3 SBIP: A Stochastic Extension for BIP

The stochastic extension of BIP allows (1) to specify stochastic aspects of individual components and (2) to provide a purely stochastic semantics for the parallel composition of components through interactions and priorities.

Stochastic Variables. Syntactically, we add stochastic behaviour to atomic components in BIP by allowing the definition of probabilistic variables. Probabilistic variables x^P are attached to given distributions μ_{x^P} implemented as C functions. These variables can then be updated on transition using the attached distribution. The semantics on transitions is thus fully stochastic. We first define atomic components and interaction between them in SBIP, and then define the corresponding stochastic semantics.

(a) Component B in SBIP (b) Semantics of B according to SBIP

Fig. 1: Example of an abstract component B and its semantics in SBIP

Definition 4 (Atomic Component in SBIP). *An atomic component in SBIP is a transition system extended with data $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$, where $L, P, T, \{g_\tau\}_{\tau \in T}$ are defined as in Definition 1, and*

- *$X = X_D \cup X_P$, with $X_D = \{x_1, \ldots, x_n\}$ the set of deterministic variables and $X_P = \{x_1^P, \ldots, x_m^P\}$ the set of probabilistic variables.*
- *For each $\tau \in T$, the update function $X' = f_\tau(X)$ is a pair $(X'_D = f_\tau^D(X), R_\tau)$ where $X'_D = f_\tau^D(X)$ is an update relation for deterministic variables and $R_\tau \subseteq X_P$ is the set of probabilistic variables that will be updated using their attached distributions. Remark that the current value of the probabilistic variables can be used in the update of deterministic variables.*

In the following, given a valuation v_X of all the variables in X, we will denote by v_Y the projection of v_X on a subset of variables $Y \subseteq X$. When clear from the context, we will denote by v_y the valuation of variable $y \in X$ in v_X.

Some transitions in the associated semantics are thus probabilistic. As an example, consider an atomic component B with a transition τ that goes from a location l to a location l' using port p and updates a probabilistic variable x^P with the distribution μ_{x^P} over the domain $\mathbf{x^P}$. In the associated semantics, assuming the initial value of x^P is v_{x^P}, there will be several transitions from state (l, v_{x^P}) to states (l', v'_{x^P}) for all $v'_{x^P} \in \mathbf{x^P}$. According to the definition of probabilistic variables, the probability of taking transition $(l, v_{x^P}) \xrightarrow{p} (l', v'_{x^P})$ will then be $\mu_{x^P}(v'_{x^P})$. This example is illustrated in Figure 1. When several probabilistic variables are updated, the resulting distribution on transitions will be the product of the distributions associated to each variables. Since these distributions are fixed from the declaration of the variables, they can be considered independent, ensuring the correctness of our construction. The syntactic definitions of interactions and composition are adapted from BIP in the same manner. For the sake of simplicity, we restrict data transfer functions on interactions to be deterministic.

Purely Stochastic Semantics. Adapting the semantics of an atomic component in BIP as presented in Definition 2 to atomic components with probabilistic variables leads to transition systems that combine both stochastic and non-

deterministic aspects. Indeed, even if atomic transitions are either purely deterministic or purely stochastic, several interactions can be enabled in a given system state. In this case, the choice between these potential transitions is non-deterministic. In order to produce a purely stochastic semantics for components defined in SBIP, we thus propose to resolve any non-deterministic choice left after applying the priorities by applying uniform distributions. Remark that other distributions could be used to resolve this non-determinism and that using uniform distributions is the default choice we made. In the future, we will allow users to specify a different way of resolving non-determinism.

Consider a component $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$ in SBIP. Given a state (l, v_X) in $L \times \mathbf{X}$, we denote by $\mathsf{Enabled}(l, v_X)$ the set of transitions in T that are enabled in state (l, v_X), i.e. transitions $\tau = (l, p, l') \in T$ such that $g_\tau(v_X)$ is satisfied. Since priorities only intervene at the level of interactions, the semantics of a single component does not take them into account. Remark that the set $\mathsf{Enabled}(l, v_X)$ may have a cardinal greater than 1. This is the only source of non-determinism in the component. In the semantics of B, instead of non-deterministically choosing between transitions in $\mathsf{Enabled}(l, v_X)$, we will choose probabilistically using a uniform distribution. Formally:

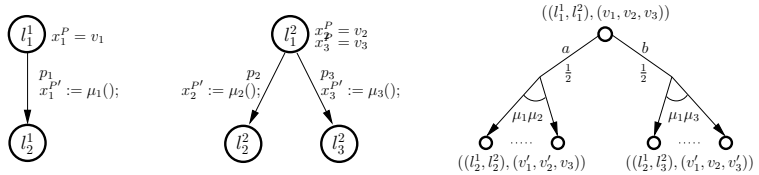
Definition 5 (Semantics of a single component in SBIP). *The semantics of $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$ in SBIP is a probabilistic transition system (Q, P, T_0) such that $Q = L \times \mathbf{X}$ and T_0 is the set of probabilistic transitions of the form $((l, v_X), p, (l', v'_X))$ for some $\tau = (l, p, l') \in \mathsf{Enabled}(l, v_X)$ such that $v'_{X_D} = f^D_\tau(v_X)$, and for all $y \in X_P \setminus R_\tau$, $v'_y = v_y$.*

In a state (l, v_X), the probability of taking a transition $(l, v_X) \xrightarrow{p} (l', v'_X)$ is the following:

$$\frac{1}{|\mathsf{Enabled}(l, v_X)|} \left[\sum_{\substack{\{\tau \in \mathsf{Enabled}(l, v_X) \\ s.t.\ \tau = (l,p,l')\}}} \left(\prod_{y \in R_\tau} \mu_y(v'_y) \right) \right].$$

The probability of taking transition $(l, v_X) \xrightarrow{p} (l', v'_X)$ is computed as follows. For each transition $\tau = (l, p, l') \in \mathsf{Enabled}(l, v_X)$ such that $v'_{X_D} = f^D_\tau(v_X)$ and for each $y \in X_P \setminus R_\tau$, $v'_y = v_y$, the probability of reaching state (l', v'_X) is $\prod_{y \in R_\tau} \mu_y(v'_y)$. Since there may be several such transitions, we take the sum of their probabilities and normalize by multiplying with $\frac{1}{|\mathsf{Enabled}(l, v_X)|}$.

Stochastic Semantics for Composing Components. When considering a system with n components in SBIP $B_i = (L_i, P_i, T_i, X_i, \{g_\tau\}_{\tau \in T_i}, \{f_\tau\}_{\tau \in T_i})$ and a set of interactions γ, the construction of the product component $B = \gamma(B_1, \ldots, B_n)$ is defined as in BIP. The resulting semantics is given by Definition 5 above, where $\mathsf{Enabled}(l, v_X)$ now represents the set of *interactions* enabled in global state (l, v_X) that are maximal with respect to priorities. By construction, it follows that the semantics of any (composite) system in SBIP is purely stochastic.



(a) Component B_1 in SBIP.

(b) Component B_2 in SBIP.

(c) Semantics of $\gamma(B_1, B_2)$ according to SBIP, with $\gamma = \{a = \{p_1, p_2\}, b = \{p_1, p_3\}\}$.

Fig. 2: Illustration of the purely stochastic semantics of composition in SBIP.

Example 1. Consider SBIP components B_1 and B_2 given in Figures 2a and 2b. B_1 has a single probabilistic variable x_1^P, to which is attached distribution μ_1 and a single transition from location l_1^1 to location l_2^1 using port p_1, where x_1 is updated. In location l_1^1, the variable x_1^P is assumed to have value v_1. B_2 has two probabilistic variables x_2^P and x_3^P, to which are attached distributions μ_2 and μ_3 respectively. B_2 admits two transitions: a transition from location l_1^2 to location l_2^2 using port p_2, where x_2 is updated, and a transition from location l_1^2 to location l_3^2 using port p_3, where x_3 is updated. In location l_1^2, the variables x_2^P and x_3^P are assumed to have values v_2 and v_3 respectively. Let $\gamma = \{a = \{p_1, p_2\}, b = \{p_1, p_3\}\}$ be a set of interactions such that interactions a and b have the same priority. The semantics of the composition $\gamma(B_1, B_2)$ is given in Figure 2c. In state $((l_1^1, l_1^2), (v_1, v_2, v_3))$ of the composition, the nondeterminism is resolved between interactions a and b, choosing one of them with probability $1/2$. After choosing the interaction, the corresponding transition is taken, updating the corresponding probabilistic variables with the associated distributions. Remark that this gives rise to a single purely stochastic transition. As an example, the probability of going to state $((l_2^1, l_2^2), (v_1', v_2', v_3))$ with interaction a is $1/2 \cdot \mu_1(v_1') \cdot \mu_2(v_2')$, while the probability of going to state $((l_2^1, l_3^2), (v_1', v_2, v_3'))$ with interaction b is $1/2 \cdot \mu_1(v_1') \cdot \mu_3(v_3')$.

An execution π of a BIP model is a sequence of states that can be generated from an initial state by following a sequence of (probabilistic) transitions. From the above, one easily sees that the semantics of any SBIP (composite) system has the structure of a discrete Markov chain. Consequently, one can define a probability measure μ on its set of executions in the usual way [22].

4 SMC approach and implementation

In this section, we present Probabilistic Bounded Linear Temporal Logic (PBLTL), a formalism for describing stochastic temporal properties. We then introduce a model checking procedure for this logic and discuss its implementation in SBIP. We first recap Bounded Linear Temporal Logic and then define its

probabilistic extension. The Bounded LTL formulas that can be defined from a set of atomic propositions $\mathbb{B}$ are the following.

- $\mathbf{T}$, $\mathbf{F}$, p, $\neg p$, for all $p \in \mathbb{B}$;
- $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, where ϕ_1 and ϕ_2 are BLTL formulas;
- $\bigcirc\phi_1$, $\phi_1 \mathcal{U}^t \phi_2$, where ϕ_1 and ϕ_2 are BLTL formulas, and t is a positive integer.

As usual, $\Diamond^t\phi = \mathbf{T}\mathcal{U}^t\phi$ and $\Box^t\phi = \neg(\mathbf{T}\mathcal{U}^t(\neg\phi))$. A Probabilistic BLTL formula is a BLTL formula preceded by a probabilistic operator P.

The semantics of a BLTL formula is defined with respect to an execution $\pi = s_0 s_1 \ldots$ in the usual way [7]. Roughly speaking, an execution $\pi = s_0 s_1 \ldots$ satisfies $\bigcirc\phi_1$, which we denote $\pi \models \bigcirc\phi_1$, if state s_1 satisfies ϕ_1. The execution π satisfies $\phi_1 \mathcal{U}^t \phi_2$ iff there exists a state s_i with $i \leq t$ that satisfies ϕ_2 and all the states in the prefix from s_0 to s_{i-1} satisfy ϕ_1.

Definition 6. *A SBIP system B satisfies the PBLTL formula $\psi = P_{\geq\theta}\phi$ iff $\mu\{\pi \mid \pi \models \phi\} \geq \theta$, where π are executions of B and μ is its underlying probability measure.*

4.1 Statistical Model Checking

Runtime verification (RV) [10, 8, 24] refers to a series of techniques whose main objective is to instrument the specification of a system (code, ...) in order to dissprove potentially complex properties at the execution level. The main problem of the runtime verification approach is that it does not permit to assess the overall correctness of the entire system.

Statistical model checking (SMC) [4, 28, 25] extends runtime verification capabilities by exploiting statistical algorithms in order to get some evidence that a given system satisfies some property.

We now present a model checking procedure to decide whether a given SBIP system B satisfies a property ψ. Consider an SBIP system B and a BLTL property ϕ. *Statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions: (1) **Qualitative:** is the probability for B to satisfy ϕ greater or equal to a certain threshold θ? and (2) **Quantitative:** what is the probability for B to satisfy ϕ? Both those questions can serve to decide a PBLTL property.

The main approaches [28, 25] proposed to answer the qualitative question are based on *hypothesis testing*. Let p be the probability of $B \models \phi$, to determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds is less or equal to α (respectively, β). Since it impossible to ensure a low probability for both types of errors simultaneously (see [28] for details), a solution is to use an *indifference region* $[p_1, p_0]$ (with θ in $[p_1, p_0]$) and to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. Several hypothesis testing algorithms

exist in the literature. Younes[28] proposed a logarithmic based algorithm that given p_0, p_1, α and β implements the *Sequential Ratio Testing Procedure* (*SPRT*) (see [26] for details). When one has to test $\theta \geq 1$ or $\theta \geq 0$, it is however better to use *Single Sampling Plan* (*SSP*) (see [28, 4, 25] for details) that is another algorithm whose number of simulations is pre-computed in advance. In general, this number is higher than the one needed by *SPRT*, but is known to be optimal for the above mentioned values. More details about hypothesis testing algorithms and a comparison between *SSP* and *SPRT* can be found in [4].

In [11, 21] Peyronnet et al. propose an estimation procedure (*PESTIMATION*) to compute the probability p for B to satisfy ϕ. Given a *precision* δ, Peyronnet's procedure computes a value for p' such that $|p' - p| \leq \delta$ with *confidence* $1 - \alpha$. The procedure is based on the *Chernoff-Hoeffding bound* [12].

The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer. This number may change from system to system and can only be estimated (see [28] for an explanation). However, some generalities are known. For the qualitative case, it is known that, except for some situations, *SPRT* is always faster than *SSP*. *PESTIMATION* can also be used to solve the qualitative problem, but it is always slower than *SSP* [28]. If θ is unknown, then a good strategy is to estimate it using *PESTIMATION* with a low confidence and then validate the result with *SPRT* and a strong confidence.

4.2 The SBIP tool

The SBIP tool implements the statistical algorithms described above, namely, *SSP*, *SPRT*, and *PESTIMATION* for SBIP systems. Figure 3 shows the tool structure and execution flow. SBIP takes as inputs a system written in SBIP, a PBLTL property to check, and a series of confidence parameters needed by the statistical test. Then, the tool creates an executable model and a monitor for the property under verification. From there, it will trigger the stochastic BIP engine to generate execution traces (Sampling) which are iteratively monitored. This procedure is repeated until a decision can be taken by the SMC core. As our approach relies on SMC, we are guaranteed that the procedure will eventually terminate.

Due to SMC restrictions, the properties must be evaluated on bounded executions. Here, we restrict to BLTL. We note that the monitoring procedure is an implementation of the work proposed in[10].

5 Case studies

While still at prototype level, SBIP has been already applied to several case studies coming from serious industrial applications.

5.1 Accuracy of Clock Synchronization Protocol IEEE.1588

Model Description. The case study concerns a clock synchronization protocol running within a distributed heterogeneous communication system (HCS) [1].

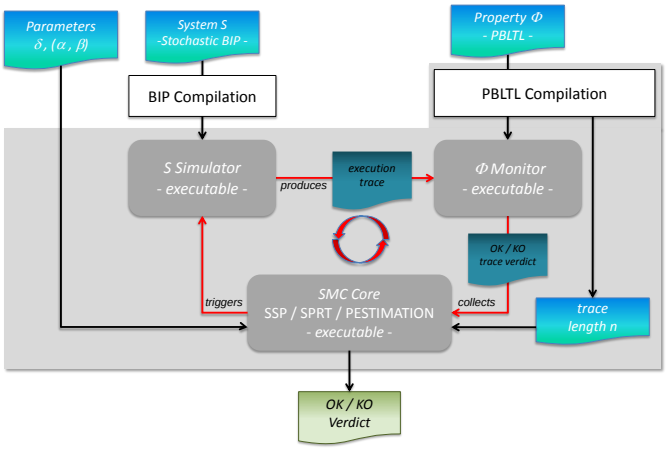


Fig. 3: SBIP tool architecture and work flow.

This protocol allows to synchronize the clocks of various devices with the one of a designated server. It is important that this synchronization occurs properly, i.e., that the difference between the clock of the server and the one of any device is bounded by a small constant.

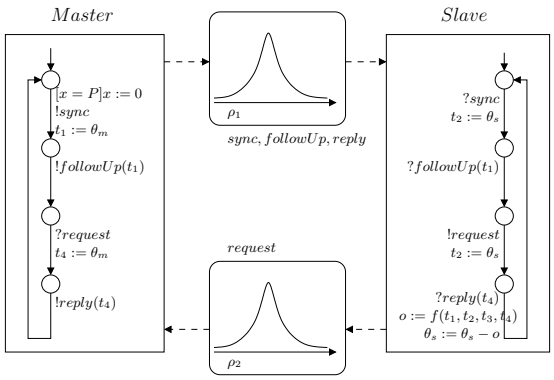


Fig. 4: PTP Stochastic Model.

To verify such property, we build the stochastic model depicted in Figure 4. This model is composed by two deterministic components namely *Master*, *Slave* and two communication channels. In the PTP model, the time of the master process is represented by the clock variable θ_m. This is considered the reference time and is used to synchronize the time of the slave clock, represented by the clock variable θ_s. The synchronization works by messages exchange between

the server and a device where each saves the message reception time $(t_i)_{i=1,4}$ w.r.t. its local clock. In termination, the slave computes the offset between its time and the master time and updates its clock accordingly. Communication channels have been modeled using stochastic components. These components model communication delays over network w.r.t empirical distributions obtained by simulating a detailed HCS model.

The accuracy of the synchronization is defined by the absolute value of the difference between the master and slave clocks $|\theta_m - \theta_s|$, during the lifetime of the system we consider (in this case, 1000 steps). Our aim is to verify the satisfaction of the formula $\phi = \Box^{1000}(|\theta_m - \theta_s| \leq \Delta)$ for arbitrary fixed non-negative Δ.

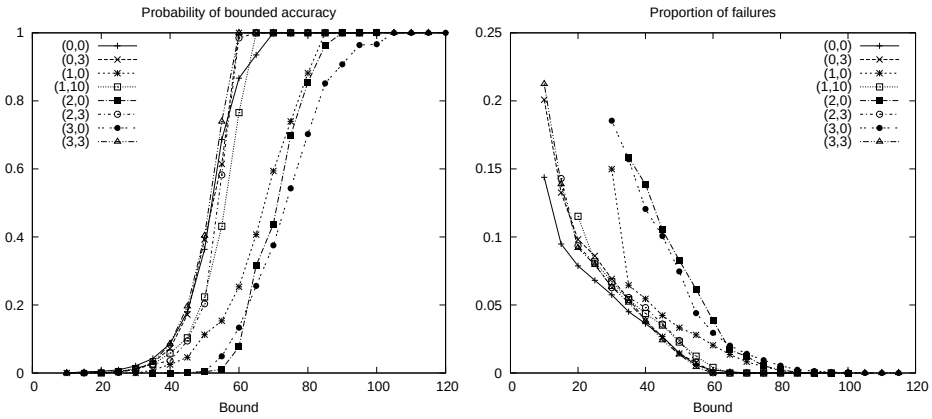
Experiments and results. Two types of experiments are conducted. The first one is concerned with the bounded accuracy property ϕ. In the second one, we study average failure per execution for a given bound.

Property 1: Synchronization. To estimate the best accuracy bound, we have computed, for each device, the probability for synchronization to occur properly for values of Δ between $10\mu s$ and $120\mu s$. Figure 5a gives the results of the probability of satisfying the bounded accuracy property ϕ as a function of the bound Δ. The figure shows that the smallest bound which ensures synchronization for any device is $105\mu s$ (for Device $(3,0)$). However, devices $(0,3)$ and $(3,3)$ already satisfy the property ϕ with probability 1 for $\Delta = 60\mu s$. For this experiments, we have used SPRT and SSP jointly with PESTIMATION for a higher degree of confidence. The results, which are presented in Table 1 for Device $(0,0)$, show that SPRT is faster than SSP and PESTIMATION.

Precision	10^{-1}		10^{-2}		10^{-3}	
Confidence	10^{-5}	10^{-10}	10^{-5}	10^{-10}	10^{-5}	10^{-10}
PESTIMATION	4883 $17s$	9488 $34s$	488243 $29m$	948760 $56m$	48824291 $> 3h$	94875993 $> 3h$
SSP	1604 $10s$	3579 $22s$	161986 $13m$	368633 $36m$	16949867 $> 3h$	32792577 $> 3h$
SPRT	316 $2s$	1176 $7s$	12211 $53s$	22870 $1m38s$	148264 $11m$	311368 $31m$

Table 1: Number of simulations / Amount of time required for PESTIMATION, SSP and SPRT.

Property 2: Average failure. In the second experiment, we try to quantify the average and worst number of failures in synchronization that occur *per simulation* when working with smaller bounds. Our goal is to study the possibility of using such bounds. For a given simulation, the *proportion of failures* is obtained



(a) Probability of satisfying bounded accuracy.

(b) Average proportion of failures.

Fig. 5: Probability of satisfying the bounded accuracy property and average proportion of failures as functions of the bound Δ.

by dividing the number of failures by the number of rounds of PTP. We will now estimate, for a simulation of 1000 steps (66 rounds of the PTP), the average value for this proportion. To this purpose, we have measured for each device this proportion on 1199 simulations with a different synchronization bounds Δ between $10\mu s$ and $120\mu s$. Figures 5b gives the average proportion of failure as a function of the bound.

5.2 Playout Buffer Underflow in MPEG2 Player

In multimedia literature [27], it has been shown that some quality degradation is tolerable when playing MPEG2-coded video. In fact, a loss under two consecutive frames within a second can be accepted. In this example, we want to check that an MPEG2 player implementation guarantees this QoS property.

Model Description. We illustrate the multimedia player set-up that has been modeled using the stochastic BIP framework. The designed model captures the stochastic system aspects that are, the macro-blocks arrival time to the input buffer and the their processing time.

The stochastic system model is shown in Figure 6. It consists of three functional components namely *Generator*, *Processor*, and *Player*. In addition to these, the buffers between the above functional components are modeled by explicit *buffer* components, namely *Input buffer* and *Playout buffer*. The transfer of the macro-blocks between the functional blocks and the buffers are described using interactions.

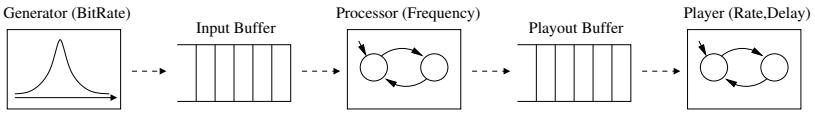
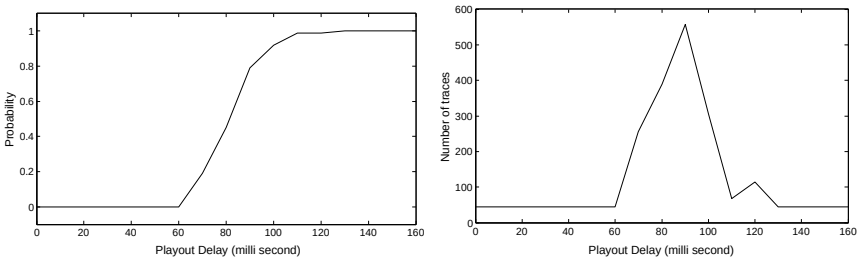


Fig. 6: MPEG2 stochastic Model.

The *Generator* is a stochastic component which models macro-blocks production based on a probabilistic distribution. It generates an MPEG2-coded stream with respect to a fixed Group-of-Pictures (GOP) pattern [18, 19] and simulates the arrival time of macro-blocks to the *input buffer*. The *Processor* reads them sequentially, decodes them and write them to the *Playout buffer*. The *Player* starts to read macro-blocks from the *Playout buffer* after a defined initial delay namely *Playout Delay*. Once this delay ends, the consumption is performed periodically with respect to a fixed consumption rate. Each period, the *Player* sends a request of N macro-blocks to the *Playout buffer*, where $N = 1$ the first time. Then it gets a response of M macro-blocks, where $0 \leq M \leq N$. We say that we have an underflow if $M < N$. In this case, the next request N will be $(N - M) + 1$. That is, the player will try to read all the missed macro-blocks.

Experiments and results. To check the described model with respect to the desired QoS property, we used the SBIP tool. The BLTL specification of the QoS property to check is $\phi = \Box^{1500000}(\neg fail)$, where *fail* denotes a failure state condition corresponding to the underflow of two consecutive frames within a second.



(a) Probability of satisfying the QoS property.

(b) Number of used traces.

Fig. 7: Illustration of results.

The obtained results are shown in Figure 7a where we can see that until $60ms$, the probability of satisfying ϕ is 0, that is, for small *Playout delays*, the number of underflow exceeds 2 frames. For delays between 70 and 100, the

probability of satisfying the QoS property increases to attain 1 for high *Playout delays* ($\geq 110ms$). Figure 7b shows the amount of needed traces for each *Playout delay* where the average simulation time is about 8 seconds.

6 Conclusion and future work

Stochastic systems can also be analyzed with a pure stochastic model checking approach. While there is no clear winner, SMC is often more efficient in terms of memory and time consumption [13]. The above experiments are out of scope of stochastic model checking. Also, there are properties such as clock drift in Clock Synchronization Protocols (see [1]) that could not have been analyzed with a pure formal approach. The PRISM toolset [20] also incorporates a stochastic model checking engine. However, it can only be applied to those systems whose individual components are purely stochastic. Moreover, probability distributions are described in a very simple and restrictive language, while we can use the full fledged C to describe complex distributions. Nevertheless, we have observed that PRISM can be faster than our tool on various case studies such as those where the same process is repeated a certain number of times. Solutions to considerably enhance the efficiency of SMC in particular cases have recently been developed [14], but have not yet been implemented in SBIP. In a recent work [6], it has been proposed to use partial order to solve non-determinism when applying SMC (which rarely works). In SBIP, the order is directly given in the design through priorities specified by the user.

We shall continue the development by implementing new heuristics to speed up simulation and to reduce their number. We shall also implement an extension of the stochastic abstraction principle from [1] that allows to compute automatically a small stochastic abstraction from a huge concrete system.

References

1. A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. In *FORTE*, volume 6117 of *LNCS*, pages 32–46. Springer, 2010.
2. A. Basu, S. Bensalem, M. Bozga, B. Delahaye, A. Legay, and E. Siffakis. Verification of an afdx infrastructure using simulations and probabilities. volume 6418 of *LNCS*. Springer, 2010.
3. A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *SEFM06*, pages 3–12, September 2006.
4. S. Bensalem, B. Delahaye, and A. Legay. Statistical model checking: Present and future. In *RV*. Springer-Verlag, 2010.
5. S. Bensalem, L. Silva, A. Griesmayer, F. Ingrand, A. Legay, and R. Yan. A formal approach for incremental construction with an application to autonomous robotic systems. In *SC'11*, LNCS. Springer, 2011.
6. J. Bogdoll, L.-M. Fiorti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FORTE*, LNCS. Springer, 2011.
7. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

8. Y. Falcone, M. Jaber, T.-H. Nguyen, M. Bozga, and S. Bensalem. Runtime verification of component-based systems. In *SEFM*, pages 204–220, 2011.
9. R. Grosu and S. A. Smolka. Monte carlo model checking. In *TACAS*, volume 3440 of *LNCS*, pages 271–286. Springer, 2005.
10. K. Havelund and G. Rosu. Synthesizing monitors for safety properties. In *TACAS*, LNCS, pages 342–356. Springer, 2002.
11. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. In *VMCAI*, pages 73–84, January 2004.
12. W. Hoeffding. Probability inequalities. *Journal of the American Statistical Association*, 58:13–30, 1963.
13. D. N. Jansen, J.-P. Katoen, M.Oldenkamp, M. Stoelinga, and I. S. Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In *HVC*, volume 4899 of *LNCS*. Springer, 2007.
14. C. Jégourel, A. Legay, and S. Sedwards. Cross entropy optimisation of importance sampling parameters for statistical model checking. In *CAV*, 2012.
15. C. Jégourel, A. Legay, and S. Sedwards. A platform for high performance statistical model checking - plasma. In *TACAS*, LNCS, pages 498–503. Springer, 2012.
16. J.-P. Katoen and I. S. Zapreev. Simulation-based ctmc model checking: An empirical evaluation. In *QEST*, pages 31–40. IEEE Computer Society, 2009.
17. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker mrmc. In *QEST*, pages 167–176. IEEE Computer Society, 2009.
18. M. Krunz, R. Sass, and H. Hughes. Statistical characteristics and multiplexing of MPEG streams. In *INFOCOM*, pages 455–462, April 1995.
19. M. Krunz and S. K. Tripathi. On the characterization of VBR MPEG streams. In *SIGMETRICS*, pages 192–202, June 1997.
20. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.
21. S. Laplante, R. Lassaigne, F. Magniez, S. Peyronnet, and M. de Rougemont. Probabilistic abstraction for model checking: An approach based on property testing. *ACM TCS*, 8(4), 2007.
22. E. Parzen. *Stochastic Processes*. Holden Day, 1962.
23. D. E. Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *ATVA*, volume 5799 of *LNCS*, pages 120–134. Springer, 2009.
24. G. Rosu and S. Bensalem. Allen linear (interval) temporal logic - translation to ltl and monitor synthesis. In *CAV*, volume 4144 of *LNCS*, pages 263–277. Springer, 2006.
25. K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.
26. A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.
27. D. Wijesekera and J. Srivastava. Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications*, 3(2):127–166, July 1996.
28. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.
29. P. Zuliani, C. Baier, and E. M. Clarke. Rare-event verification for stochastic hybrid systems. In *HSCC*, pages 217–226. ACM, 2012.
30. P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *HSCC*, pages 243–252. ACM, 2010.