INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Statistical Abstraction and Model-Checking of Large Heterogeneous Systems

Ananda Basu, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS — Saddek Bensalem, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS — Marius Bozga, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS — Benoît Caillaud, INRIA / IRISA, Rennes, France — Benoît Delahaye, Université de Rennes 1 / IRISA, France — Axel Legay, INRIA / IRISA, Rennes, France

N° 7238

Mars 2010

Thèmes COM et SYM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7238--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Statistical Abstraction and Model-Checking of Large Heterogeneous Systems[*]

Ananda Basu, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS , Saddek Bensalem, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS , Marius Bozga, Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS , Benoît Caillaud, INRIA / IRISA, Rennes, France , Benoît Delahaye, Université de Rennes 1 / IRISA, France , Axel Legay, INRIA / IRISA, Rennes, France

Thèmes COM et SYM — Systèmes communicants et Systèmes symboliques
Équipe-Projet S4

Rapport de recherche n° 7238 — Mars 2010 — 29 pages

Abstract: We propose a new simulation-based technique for verifying applications running within a large heterogeneous system. Our technique starts by performing simulations of the system in order to learn the context in which the application is used. Then, it creates a stochastic abstraction for the application, which takes the context information into account. This smaller model can be verified using efficient techniques such as statistical model checking. We have applied our technique to an industrial case study: the cabin communication system of an airplane. We use the BIP toolset to model and simulate the system. We have conducted experiments to verify the clock synchronization protocol i.e., the application used to synchronize the clocks of all computing devices within the system.

Key-words: Statistical Model-Checking, BIP, Abstraction

[*] This work has been supported by the Combest EU project.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Abstraction Statistique et Model-Checking de Systèmes Hétérogènes Complexes[†]

Résumé : Nous proposons une nouvelle technique à base de simulations pour la vérification d'applications implantées dans un système hétérogène complexe. Cette technique commence oar effectuer des simulations du système dans le but d'apprendre le contexte dans lequel l'application sera utilisée. Ensuite, elle crée une abstraction stochastique pour l'application, qui prend compte de l'information venant du contexte. Ce modèle plus simple peut ensuite être vérifié en utilisant des techniques efficaces comme le model-checking statistique. Nous avons appliqué cette technique à un cas d'étude industriel: le système de communication de cabinne d'un avion. Nous utilisons BIP pour modéliser et simuler le système. Nous avons conduit des expériences dans le but de vérifier le protocole de synchronisation des horloges i.e., l'application utilisée pour synchroniser les horloges de tous les composants actifs du système.

Mots-clés : Model-Checking Statistique, BIP, Abstraction

1 Introduction

Systems integrating multiple heterogeneous distributed applications communicating over a shared network are typical in various sensitive domains such as aeronautic or automotive embedded systems. Verifying the correctness of a particular application inside such a system is known to be a challenging task, which is often beyond the scope of existing exhaustive validation techniques. The main difficulty comes from network communication which makes all applications interfering and therefore forces exploration of the full state-space of the system.

Statistical Model Checking [7, 12, 14] has recently been proposed as an alternative to avoid an exhaustive exploration of the state-space of the model. The core idea of the approach is to conduct some simulations of the system and then use statistical results in order to decide whether the system satisfies the property or not. Statistical model checking techniques can also be used to estimate the probability that a system satisfies a given property [7, 6]. Of course, in contrast with an exhaustive approach, a simulation-based solution does not guarantee a correct result. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory and time intensive than exhaustive ones, and are sometimes the only option [15, 9]. Statistical model checking is widely accepted in various research areas such as systems biology [5, 10] or software engineering, in particular for industrial applications. There are several reasons for this success. First, it is very simple to implement, understand and use. Second, it does not require extra modeling or specification effort, but simply an operational model of the system, that can be simulated and checked against state-based properties. Third, it allows model-checking of properties [4] that cannot be expressed in classical temporal logics. Nevertheless, statistical-model checking still suffers from the system's complexity. In particular, for the case of heterogeneous systems, the number of components and their interactions are limiting factors on the number and length of simulations that can be conducted and hence on the accuracy of the statistical estimates.

We propose to exploit the structure of the system in order to increase the efficiency of the verification process. The idea is conceptually simple: instead of performing an analysis of the entire system, we propose to analyze each application separately, but under some particular context/execution environment. This context is a *stochastic abstraction* that represents the interactions with other applications running within the system and sharing the computation and communication resources. We propose to build such a context automatically by simulating the system and learning the probability distributions of key characteristics impacting the functionality of the given application.

The overall contribution of this paper is an application of the above method on an industrial case study, the *heterogeneous communication system* (HCS for short) deployed for cabin communication in a civil airplane. HCS is an heterogeneous system providing entertainment services (e.g.: audio/video on passengers demand) as well as administrative services (e.g.: cabin illumination, control, audio announcements), which are implemented as distributed applications running in parallel, across various devices within the plane and communicating through a common Ethernet-based network. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timing and synchronization constraints. An important requirement, which will be studied in this paper, is the *accuracy of clock synchronization* between different devices. This latter property states that the difference between the clocks of any two devices should be bounded by a small constant, which is provided by the user and depends on his needs. Hence, one must be capable of computing the smallest bound for which synchronization occurs and compare it with the bound expected by the user. Unfortunately, due to the large number of heterogeneous components that constitute the system, deriving such a bound manually from the textual specification is an unfeasible task. In this paper, we propose a formal approach that consists in building a formal model of the HCS, then applying simulation-based algorithms to this model in order to deduce the smallest value of the bound for which synchronization occurs. We start with a fixed value of the bound and check whether synchronization occurs. If yes, then we make sure that this is the best one. If no, we restart the experiment with a new value.

At the top of our approach, there is a tool that should be capable of modeling heterogeneous systems as well as simulating their executions and the many interactions between components. In this paper, we propose to use the BIP toolset [2] for doing so. BIP supports a methodology for building systems from atomic components that communicate using shared resources. BIP also offers a powerful engine to simulate the system and can thus be combined with a statistical model checking algorithm in order to verify properties. Our first contribution is to study all the requirements for the HCS to work properly and then derive a model in BIP. The model we obtain is quite big : almost 300 atomic components (2468 description lines in BIP code, 9018 auto-generated lines in C), 245 clocks, and 2^{300} states. Our second contribution is to study the accuracy of clock synchronization between several devices of the HCS. In HCS the clock synchronization is ensured by the *Precision Time Protocol* (PTP for short) [1], and the challenge is to guarantee that PTP maintains the difference between a master clock (running on a designated server within the system) and all the slave clocks (running on other devices) under some bound. Since this bound cannot be pre-computed, we have to verify the system for various values of the bound until we find a suitable one. Unfortunately, the full system is too big to be analyzed with classical exhaustive verification techniques. A solution could be to remove all the information that is not related to the devices under consideration. This is in fact not correct as the behavior of the PTP protocol is influenced by the many other applications running in parallel within the heterogeneous system. Our solution to this state-space explosion problem is in two steps (1) we will build a stochastic abstraction for a part of the PTP application between the server and a given device; the stochastic part will be used to model the general context in which PTP is used, (2) we will apply statistical model checking on the resulting model.

Thanks to this approach, we have been able to derive precise bounds that guarantee proper synchronization for all the devices of the system. We also computed the probability of satisfying the property for smaller values of the bound, i.e., bounds that do not satisfy the synchronization property with probability 1. Being able to provide such information is of clear importance, especially when the best bound is too high with respect to the user's requirements. We have observed that the values we obtained strongly depend on the position of the device in the network. We also estimated the average and worst proportion of failures per simulation for bounds that are smaller than the one that guarantees synchronization. Checking this latter property has been made easy because BIP allows us to reason on one execution at a time. Finally, we have also considered the influence of clock drift on the synchronisation results. The experiments highlight the generality of our technique, which could be applied to other versions of the HCS as well as to other heterogeneous applications.

2 An Overview of Statistical Model Checking

Consider a stochastic system $\mathcal{S}$ and a 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 that $\mathcal{S}$ satisfies ϕ greater or equal to a certain threshold? and (2) **Quantitative :** What is the probability that $\mathcal{S}$ satisfies ϕ? Contrary to numerical approaches, the answer is given up to some correctness precision. In the rest of the section, we overview several statistical model checking techniques. Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise.

2.1 Qualitative Answer using Statistical Model Checking

The main approaches [14, 12] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\phi)$, 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, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β). A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [14] 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 $H_1 : p \leq p_1$. We now sketch two hypothesis testing algorithms.

Single Sampling Plan. To test H_0 against H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan (SSP in short)*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. Clearly, this number has to be greater if α and β are smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases [14]. In his thesis [14], Younes proposes a binary search based algorithm that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

Sequential probability ratio test. The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n-m$ other simulations. To overcome this problem, one can use the *sequential probability ratio test (SPRT in short)* proposed by Wald [13]. The approach is briefly described below.

In SPRT, one has to choose two values A and B $(A > B)$ that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1[13]. This has the advantage of minimizing the number of simulations. In his thesis [14], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure.

2.2 Quantitative Answer using Statistical Model Checking

In [7, 11] Peyronnet et al. propose an estimation procedure to compute the probability p for $\mathcal{S}$ to satisfy ϕ. Given a *precision* δ, Peyronnet's procedure, which we call PESTIMATION, computes a value for p' such that $|p'-p| \leq \delta$ with *confidence* $1-\alpha$. The procedure is based on the *Chernoff-Hoeffding bound* [8]. Let $B_1 \ldots B_m$ be m discrete random variables with a Bernoulli distribution of parameter p associated with m simulations of the system. Recall that the outcome for each of the B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. Let $p' = (\sum_{i=1}^{m} b_i)/m$, then Chernoff-Hoeffding bound [8] gives $Pr(|p'-p| > \delta) < 2e^{-\frac{m\delta^2}{4}}$. As a consequence, if we take $m \geq \frac{4}{\delta^2}\log(\frac{2}{\alpha})$, then $Pr(|p'-p| \leq \delta) \geq 1-\alpha$. Observe that if the value p' returned by PESTIMATION is such that $p' \geq \theta - \delta$, then $\mathcal{S} \models Pr_{\geq\theta}$ with confidence $1-\alpha$.

2.3 Playing with Statistical Model Checking Algorithms

The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer. This number may change from executions to executions and can only be estimated (see [14] 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. When $\theta = 1$ (resp. $\theta = 0$) SPRT degenerates to SSP; this is not problematic since SSP is known to be optimal for such values. PESTIMATION can also be used to solve the qualitative problem, but it is always slower than SSP [14]. 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.

3 Validation Method and the BIP Toolset

Consider a system consisting of a set of distributed applications running on several computers and exchanging messages on a shared network infrastructure. Assume also that network communication is subject to given bandwidth restrictions as well as to routing and scheduling policies applied on network elements. Our method attempts to reduce the complexity of validation of a particular application of such system by decoupling the timing analysis of the network and functional analysis of each application.

We start by constructing a model of the whole system. This model must be executable, i.e., it should be possible to obtain execution traces, annotated with timing information. For a chosen application, we then learn the probability distribution laws of its message delays by simulating the entire system. The method then constructs a reduced stochastic model by combining the application model where the delays are defined according to the laws identified at the previous step. Finally, the method applies statistical model-checking on the resulting stochastic model.

Our models are specified within the *Behaviour-Interaction-Priority* (BIP for short) framework [2]. BIP is a component-based framework for construction, implementation and analysis of systems composed of heterogeneous components. In particular, BIP fulfills all the requirements of the method suggested above, that are, models constructed in BIP are operational and can be thoroughly simulated. BIP models can easily integrate timing constraints, which are represented with discrete clocks. Probabilistic behaviour can also be added by using external C functions.

The BIP framework is implemented within the BIP toolset [3], which includes a rich set of tools for modeling, execution, analysis (both static and on-the-fly) and static transformations of BIP models. It provides a dedicated programming language for describing BIP models. The front-end tools allow editing and parsing of BIP programs, and generating an intermediate model, followed by code generation (in C) for execution and analysis on a dedicated middleware platform. The platform also offers connections to external analysis tools. A more complete description of BIP can be found in the appendix.

4 Case Study: Heterogeneous Communication System

The case study concerns a distributed heterogeneous communication system (HCS) providing an all electronic communication infrastructure to be deployed, typically for cabin communication in airplanes or for building automation. The HCS system contains various devices such as sensors (video camera, smoke detector, temperature, pressure, etc.) and actuators (loudspeakers, light switches, temperature control, signs, etc.) connected through a wired communication network to a common server. The server runs a set of services to monitor the sensors and to control the actuators. The devices are connected to the server using network access controllers (NAC) as shown for an example architecture in Figure 1. An extended star-like HCS architecture with several daisy chains of NACs and devices is presented in appendix B.

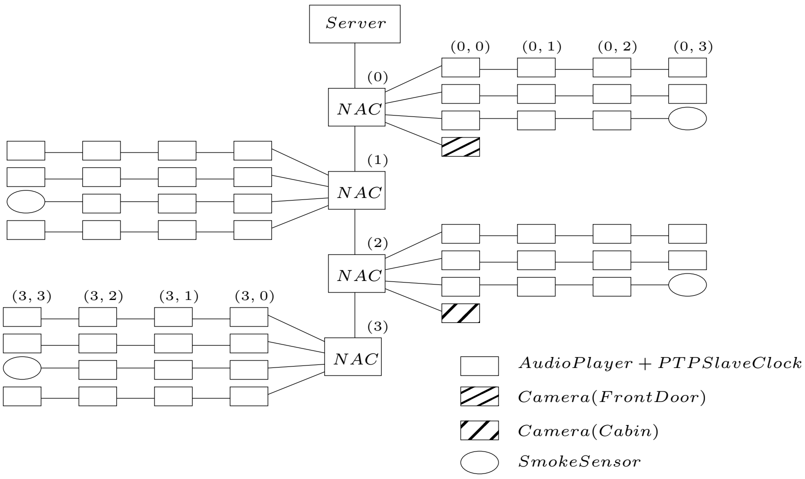


Figure 1: HCS Example Model.

The system-wide HCS architecture is highly heterogeneous. It includes hardware components and software applications ensuring functions with different characteristics and degree of criticality e.g, audio streaming, device clock synchronisation, sensor monitoring, video surveillance. It also integrates different communication and management protocols between components. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timings and synchronization constraints. For example, the latency for delivering alarm signals from sensors, or for playing audio announcements should be smaller than certain predefined thresholds. Or, the accuracy of clock synchronization between different devices, should be guaranteed under the given physical implementation of the system.

The HCS case study poses challenges that require component-based design techniques, since it involves heterogeneous components and communication mechanisms, e.g. streaming based on the data-flow paradigm as well as event driven computation and interaction. Its modeling needs combination of executable and analytic models especially for performance evaluation and analysis of non-functional properties.

We have modeled an instance of the HCS system in BIP. As shown in Figure 1, the system consists of one *Server* connected to a daisy chain of four NACs, addressed $0 \cdots 3$, and several devices. Devices are connected in daisy chains with the NACs, the length of each chain being limited to four in our example. For simplicity, devices are addressed (i, j), where i is the address of the NAC and j is the address of the device. The model contains three types of devices, namely *Audio Player*, *Video Camera* and *Smoke Sensor*. The devices connected to NAC(0) and NAC(2) have similar topology. The first two daisy-chains consist of only *Audio Player* devices. The third daisy-chain ends with a *Smoke Sensor*, and the fourth daisy-chain consists of just one *Video Camera*. The devices connected to NAC(1) and NAC(3) have exactly the same topology, consisting of several *Audio Player* and one *Smoke Sensor* devices.

The system depicted in Figure 1 contains 58 devices in total. The BIP model contains 297 atomic components, 245 clocks, and its state-space is of order 2^{3000}. The size of the BIP code for describing the system is 2468 lines, which is translated to 9018 lines in C. A description of the key components is given in appendix.

5 Experiments on the HCS

One of the core applications of the HCS case study is the PTP protocol, which allows the synchronization of the clocks of the various devices with the one of the 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. Studying this problem is the subject of this section. Since the BIP model for the HCS is extremely large (number of components, size of the state space, ...), there is no hope to analyse it with an exhaustive verification technique. Here, we propose to apply our stochastic abstraction. Given a specific device, we will proceed in two steps. First, we will conduct simulations on the entire system in order to learn the probability distribution on the communication delays between this device and the server. Second, we will use this information to build a stochastic abstraction of the application on which we will apply statistical model checking. We start with the stochastic abstraction for the PTP.

5.1 The Precision Time Protocol IEEE 1588

The Precision Time Protocol [1] has been defined to synchronize clocks of several computers interconnected over a network. The protocol relies on multicast communication to distribute a reference time from an accurate clock (*the master*) to all other clocks in the network (*the slaves*) combined with individual offset correction, for each slave, according to its specific round-trip communication delay to the master. The accuracy of synchronization is negatively impacted by the jitter (i.e., the variation) and the asymmetry of the communication delay between the master and the slaves. Obviously, these delay characteristics are highly dependent on the network architecture as well as on the ongoing network traffic.

We present below the abstract stochastic model of the PTP protocol between a device and the server in the HCS case study. The model consists of two (deterministic) application components respectively, the master and the slave clocks, and two probabilistic components, the media, which are abstraction of the communication network between the master and the slave. The former represent the behaviour of the protocol and are described by extended timed i/o-automata. The latter represent a random transport delay and are simply described by probabilistic distributions. Recap that randomization is used to represent the context, i.e., behaviors of other devices and influence of these behaviors on those of the master and the device under consideration.

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 as follows. Periodically, the master broadcast a *sync* message and immediately after a *followUp* message containing the time t_1 at which the *sync* message has

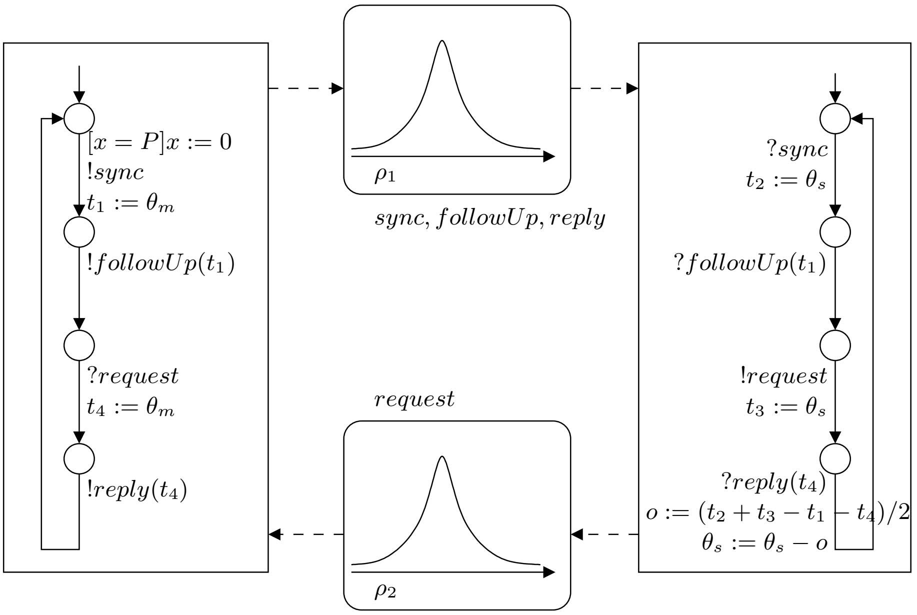


Figure 2: Abstract stochastic PTP between the server and a device.

been sent. Time t_1 is observed on the master clock θ_m. The slave records in t_2 the reception time of the *sync* message. Then, after the reception of the *followUp*, it sends a delay *request* message to the master and records its emission time t_3. Both t_2 and t_3 are observed on the slave clock θ_s. The master records on t_4 the reception time of the *request* message and sends it back to the slave on the *reply* message. Again, t_4 is observed on the master clock θ_m. Finally, upon reception of *reply*, the slave computes the offset between its time and the master time based on $(t_i)_{i=1,4}$ and updates its clock accordingly. In our model, the offset is computed differently in two different situations. In the first situation, which is depicted in Figure 2, the average delays from master to slave and back are supposed to be equal i.e., $\mu(\rho_1) = \mu(\rho_2)$. In the second situation, delays are supposed to be asymmetric, i.e., $\mu(\rho_1) \neq \mu(\rho_2)$. In this case, synchronization is improved by using an extra offset correction which compensate for the difference, more precisely, $o := (t_2 + t_3 - t_1 - t_4)/2 + (\mu(\rho_2) - \mu(\rho_1))/2$. This offset computation is an extension of the PTP specification and has been considered since it ensures better precision when delays are not symmetric (see Section 5).

Encoding the abstract model of timed i/o-automata given in Figure 2 in BIP is quite straightforward and can be done with the method presented in [2]. The distribution on the delay is implemented as a new C function in the BIP model. It is worth mentioning that, since the two i/o automata are deterministic, the full system depicted in Figure 2 is purely stochastic.

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 time. Our aim is to check the (safety) property of bounded accuracy ϕ_Δ, that is, *always* $|\theta_m - \theta_s| \leq \Delta$ for arbitrary fixed non-negative real Δ.

Finally, a simpler version of this protocol has been considered (see appendix C) and analyzed. In that study, delay components have been modeled using non-deterministic timed i/o automata as well and represent arbitrary delays bounded in some intervals $[L, U]$. It has been shown that, if the clock drift is negligible, the best accuracy $\Delta^\star$ that can be obtained using PTP is respectively $\frac{U-L}{2}$ in the symmetric case, and $\frac{U_1+U_2-L_1-L_2}{4}$ in the asymmetric case. That is, the property of bounded accuracy holds trivially iff $\Delta \geq \Delta^*$.

5.2 Model Simulations

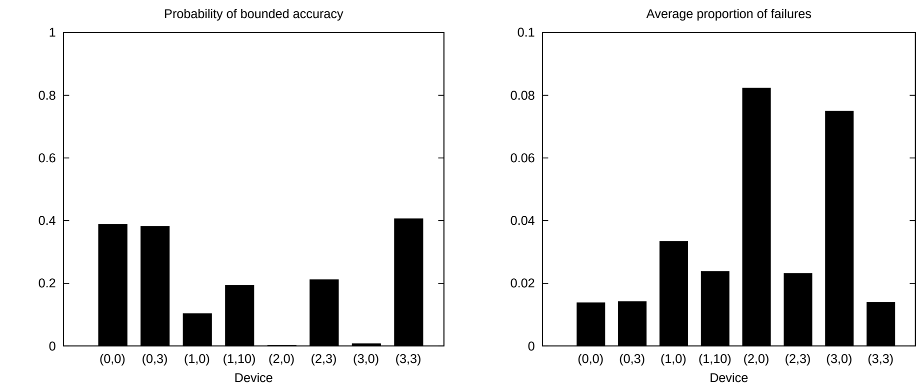
In this section, we describe our approach to learn the probability distribution over the delays. Consider the server and a given device. In a first step, we run simulations on the system and measure the end-to-end delays of all PTP messages between the selected device and the server. For example, consider the case of delay *request* messages and assume that we made 33 measures. The result will be a series of delay values and, for each value, the number of times it has been observed. As an example, delay 5 has been observed 3 times, delay 19 has been observed 30 times. The probability distribution is represented with a table of 33 cells. In our case, 3 cells of the table will contains the value 5 and 30 will contain the value 19. The BIP engine will select a value in the table following a uniform probability distribution.

According to our experiments, 2000 delay measurements are enough to obtain an accurate estimation of the probability distribution. However, for confidence reasons, we have conducted 4000 measurements. We have also observed that the value of the distribution clearly depends on the position of the device in the topology (see Appendix D for an illustration). It is worth mentioning that running one single simulation allowing 4000 measurements of the delay of PTP frames requires running the PTP protocol with an increased frequency i.e., the default PTP period (2 minutes) being far too big compared with the period for sending audio/video packets (tens of milliseconds). Therefore, we run simulations where PTP is executed once every 2 milliseconds and, we obtain 4000 measurements by simulating approximately 8 seconds of the global system lifetime. Each simulation uses microsecond time granularity and takes around 40 minutes on a Pentium 4 running under a Linux distribution.

5.3 Experiments on Precision Estimation for PTP

Three sets of experiments are conducted. The first one is concerned with the bounded accuracy property (see Section 5.1). In the second one, we study average failure per execution for a given bound. Finally, we study the influence of drift on the results.

Property 1 : Synchronization. Our objective is to compute the smallest bound Δ under which synchronization occurs properly for any device. We start with an experiment that shows that the value of the bound depends on the place of the device in the topology. For doing so, we use $\Delta = 50\mu s$ as a bound and then compute the probability for synchronization to occur properly for all the devices. In the paper, for the sake of presentation, we will only report on a sampled set of devices :$(0,0)$, $(0,3)$, $(1,0)$, $(1,10)$, $(2,0)$, $(2,3)$, $(3,0)$, $(3,3)$, but our global observations extend to any device. We use PESTIMATION with a



(a) Probability of satisfying bounded accuracy.

(b) Average proportion of failures.

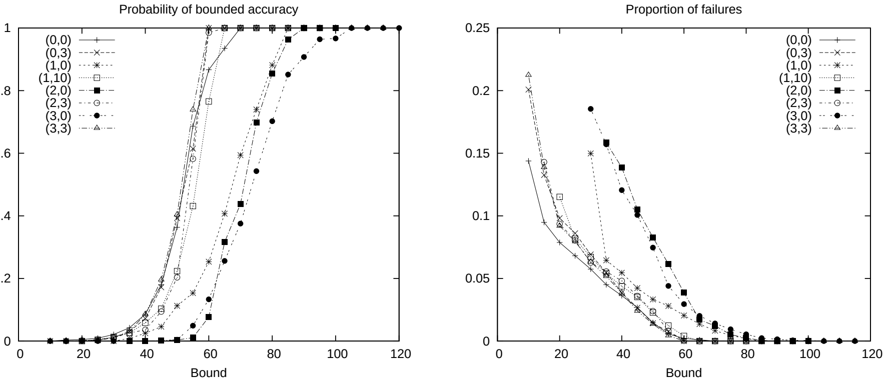
Figure 3: Probability of satisfying the bounded accuracy property and average proportion of failures for a bound $\Delta = 50\mu s$ and the asymmetric version of PTP.

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.

confidence of 0.1. The results, which are reported in Figure 3a, show that the place in the topology plays a crucial role. Device $(3, 3)$ has the best probability value and Device $(2, 0)$ has the worst one. All the results in Figure 3a have been conducted on the model with asymmetric delays. For the symmetric case, the probability values are much smaller. As an example, for Device $(0, 0)$, it decreases from 0.388 to 0.085. The above results have been obtained in less than 4 seconds. As a second experiment, we have used SPRT and SSP to validate the probability value found by PESTIMATION with 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.

Our second step was to estimate the best bound. For doing so, for each device we have repeated the previous experiments for values of Δ between $10\mu s$ and $120\mu s$. Figure 4a gives the results of the probability of satisfying the bounded accuracy property as a function of the bound Δ for the asymmetric version of PTP. The figure shows that the smallest bound which ensure synchronization for any device is $105\mu s$ (for Device $(3, 0)$). However, devices $(0, 3)$ and $(3, 3)$



(a) Probability of satisfying bounded accuracy.

(b) Average proportion of failures.

Figure 4: Probability of satisfying the bounded accuracy property and average proportion of failures as functions of the bound Δ for the asymmetric version of PTP.

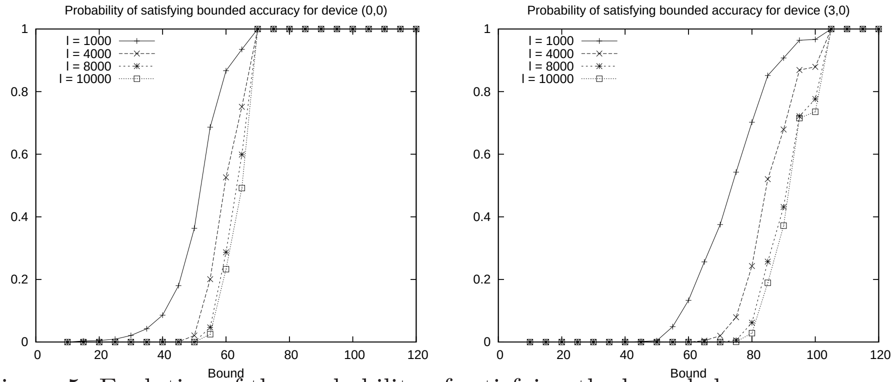
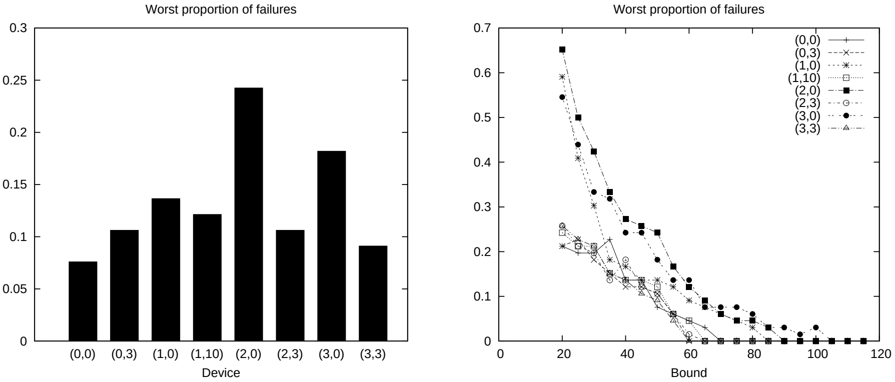


Figure 5: Evolution of the probability of satisfying the bounded accuracy property with the length of the simulations for the asymmetric version of PTP.

already satisfy the property with probability 1 for $\Delta = 60\mu s$. A comparison between SSP and PESTIMATION is given in the appendix.

The above experiments have been conducted assuming simulations of 1000 BIP interactions and 66 rounds of the PTP protocol. Since each round of the PTP takes two minutes, this also corresponds to 132 minutes of the system's life time. We now check whether the results remain the sames if we lengthen the simulations and hence system's life time. Figure 5 shows, for Devices $(0,0)$ and $(3,0)$, the probability of synchronization for various values of Δ and various length of simulations (1000, 4000, 8000 and 10000 (660 minutes of system's life time) steps). We used PESTIMATION with a precision and a confidence of 0.1. The best bounds do not change. However, the longer the simulations are, the more the probability tends to be either 0 or 1 depending on the bound.

Property 2 : Average failure. In the previous experiment, we have computed the best bound to guarantee the bounded accuracy property. It might



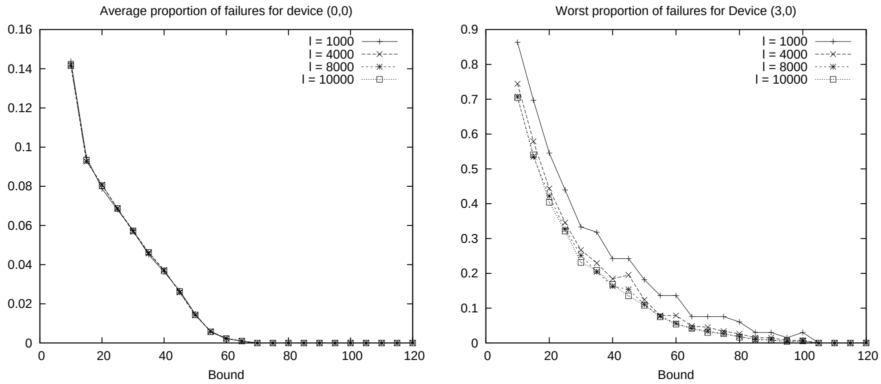
(a) Worst proportion of failures for bound $\Delta = 50\mu s$.

(b) Worst proportion of failures as a function of the bound Δ.

Figure 6: Worst proportion of failures for the industrial bound $\Delta = 50\mu s$ and as a function of the bound Δ for the asymmetric version of PTP.

be the case that the bound is too high regarding the user's requirements. In such case, using the above results, we can already report on the probability for synchronization to occur properly for smaller values of the bound. We now give a finer answer by quantifying the average and worst number of failures in synchronization that occur *per simulation* when working with smaller bounds. For a given simulation, the *proportion of failures* is obtained 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 and worst value for this proportion. To this purpose, we have measured (for each device) this proportion on 1199 simulations with a synchronization bound of $\Delta = 50\mu s$. As an example, we obtain average proportions of 0.036 and 0.014 for Device $(0, 0)$ using the symmetric and asymmetric versions of PTP respectively. As a comparison, we obtain average proportions of 0.964 and 0.075 for Device $(3, 0)$. The average proportion of failures with the bound $\Delta = 50\mu s$ and the asymmetric version of PTP is given in Figure 3b. Figure 6a presents, for the sampled devices, the worst proportion of failures using the asymmetric version of PTP. The worst value is $0, 25$, which is obtained for Device $(2, 0)$. On the other hand, the worst value is only $0, 076$ for Device $(0, 0)$. The experiment, which takes about 6 seconds per device, was then generalized to other values of the bound. Figures 4b and 6b give the average and worst proportion of failure as a function of the bound.

The above experiment gives, for several value of Δ and each device, the worst failure proportion with respect to 1199 simulations. We have also used PESTIMATION with confidence of 0.1 and precision of 0.1 to verify that this value remains the same whatever the number of simulations is. The result was then validated using SSP with precision of 10^{-3} and confidence of 10^{-10}. Each experiment took approximately two minutes. Finally, we have conducted experiments to check whether the results remain for longer simulations. Figure 7a shows that the average proportion does not change and Figure 7b shows that the worst proportion decreases when the length of the simulation increases.



(a) Average proportion of failure for Device (0,0).

(b) Worst proportion of failure for Device (3,0).

Figure 7: Evolution of the average and worst proportion of failures with the length of the simulations for the asymmetric version of PTP.

Clock Drift. We have considered a modified version of the stochastic PTP model with drifting clocks. Drift is used to model the fact that, due to the influence of the hardware, clocks of the master and the device may not progress as the same rate. In our model, drift is incorporated as follows: each time the clock of the server is increased by 1 time unit, the clock of the device is increased by $1 + \varepsilon$ time units, with $\varepsilon \in [-10^{-3}, 10^{-3}]$. Using this modified model, we have re-done the experiments of the previous sections and observed that the result remains almost the same. This is not surprising as the value of the drift significantly smaller than the communication jitter, and therefore it has less influence of the synchronization. A drift of 1 time unit has a much higher impact on the probability. As an example, for Device $(0, 0)$, it goes from a probability of $0, 387$ to a probability of $0, 007$. It is worth mentioning that exhaustive verification of a model with drifting clocks is not an easy task as it requires to deal with complex differential equations. When reasoning on one execution at a time, this problem is avoided.

References

[1] I. I. 61588. *Precision clock synchronization protocol for networked measurement and control systems*, 2004.

[2] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *SEFM06, Pune, India*, pages 3–12, September 2006.

[3] The BIP Toolset. http://www-verimag.imag.fr/∼async/bip.php.

[4] E. M. Clarke, A. Donzé, and A. Legay. Statistical model checking of mixed-analog circuits with an application to a third order delta-sigma modulator. In *HVC*, volume 5394 of *LNCS*, pages 149–163. Springer, 2008. to appear.

[5] E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay. Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In *CMSB*, volume 5307 of *LNCS*, pages 231–250. Springer, 2008.

[6] R. Grosu and S. A. Smolka. Monte carlo model checking. In *TACAS*, volume 3440 of *LNCS*, pages 271–286. Springer, 2005.

[7] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, volume 2937 of *LNCS*, pages 73–84. Springer, 2004.

[8] W. Hoeffding. Probability inequalities. *Journal of the American Statistical Association*, 58:13–30, 1963.

[9] 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.

[10] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. In *CMSB*, volume 5688 of *LNCS*, pages 218–234. Springer, 2009.

[11] S. Laplante, R. Lassaigne, F. Magniez, S. Peyronnet, and M. de Rougemont. Probabilistic abstraction for model checking: An approach based on property testing. *ACM Trans. Comput. Log.*, 8(4), 2007.

[12] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.

[13] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[14] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.

[15] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

A Appendix for Section 3 : An overview of BIP

The BIP framework, presented in [2], supports a methodology for building systems from *atomic components*. It uses *connectors*, to specify possible interaction patterns between components, and *priorities*, to select amongst possible interactions. In BIP, data and their transformations can be written directly in C.

Atomic components are finite-state automata extended with variables and ports. Ports are action names, and may be associated with variables. They are used for synchronization with other components. Control states denote locations at which the components await for synchronization. Variables are used to store local data.

We provide in Figure 8 an example of an atomic component, named *Router*, that models the behavior of a network router. It receives network packets

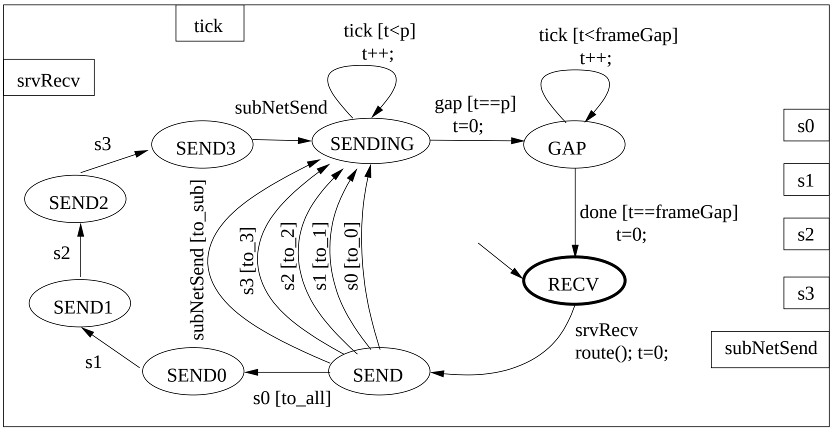


Figure 8: An atomic component: Router.

through an input port and delivers them to the respective output port(s), based on the destination address of the packets. The port *srvRecv* acts as an input port, while *s*0, *s*1, *s*2, *s*3, and *subNetSend* act as output ports. The port *tick* is used for modeling time progress and specific timing constraints. The control locations are *RECV*, *SEND*, *SEND0*, *SEND1*, *SEND2*, *SEND3*, *SENDING* and *GAP*, with *RECV* being the initial location. It also has the variables *t, p, to_0, to_1, to_2, to_3, to_sub, to_all,frame*, and parameter *frameGap*.

A transition is a step from a control location to another, guarded by a Boolean condition on the set of its variables, labeled by a port. An example transition is from the initial location *RECV* to *SEND*, which is executed when an interaction including port *srvRecv* takes place, the default guard being *true*. On execution, the internal computation step is the execution of the C routine *route()*, followed by the reset of the variable *t*.

Composite components allow defining new components from sub-components (atomic or composite). Components are connected through flat or hierarchical *connectors*, which relate ports from different sub-components. Connectors represent sets of interactions, that are, non-empty sets of ports that have to be jointly executed. They also specify guards and transfer functions for each interaction, that is, the enabling condition and the exchange of data across the ports of the interacting components.

Figure 9 shows a composite component *Server*. It contains atomic components $service_0 \cdots service_n$, *FrameReceiver*, and composite components *Classifier* and *NAC*. The *NAC* contains a *Router* and a*Classifier*. The connectors are shown by lines joining the ports of the components.

Priorities are used to select amongst simultaneously enabled interactions. They are a set of 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 the one with higher-priority can be executed.

The architecture of a generic device is shown in Figure 10.

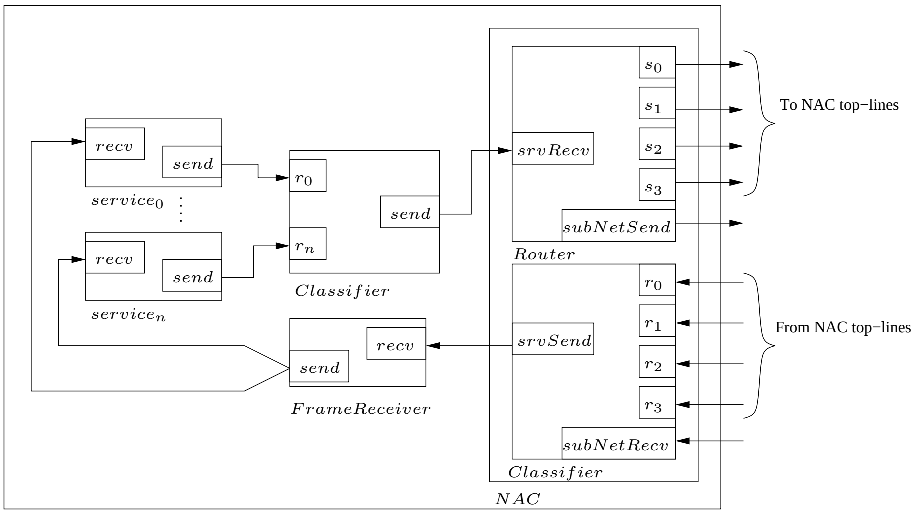


Figure 9: Composite Component: Server.

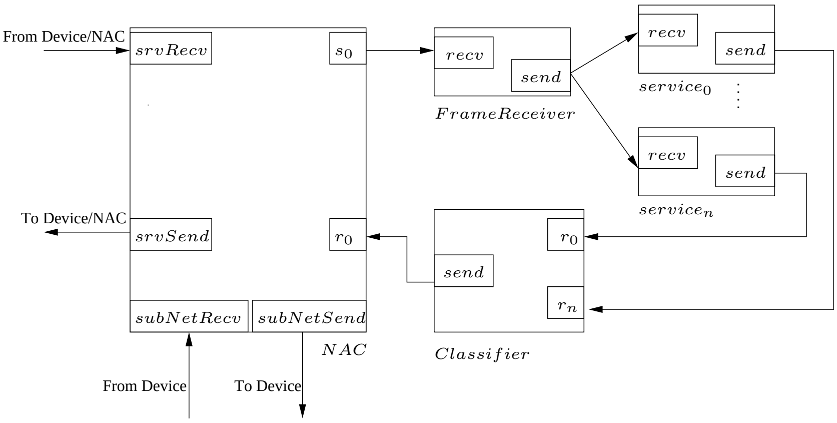


Figure 10: A device component.

B Appendix for Section 4

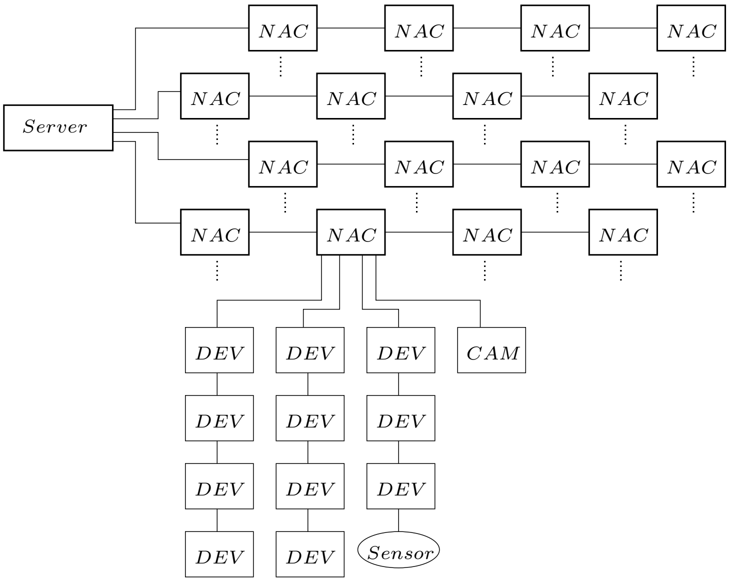


Figure 11: Heterogeneous Communication System (HCS).

Server. The server runs various applications including: 1) PTP *MasterClock*, that runs the PTP master-clock protocol between the server and the devices in order to keep the device clocks synchronized with the master-clock. The protocol exchanges PTP packets of size 512 bits between the server and the devices, and runs once every 2 minutes. 2) *AudioGenerator*, that generates audio streams to be playbacked by the *Audio Player* devices. It generates audio streams at 32kHz with 12 bit resolution (audio chunks). We have assumed that 100 audio chunks are sent in a single frame over the network, (that gives the size of an audio frame to be 1344 bits) at the rate of 33 frames per second. 3) *SmokeDetector* service that keeps track of the event packets (size 736 bits) sent from the *Smoke Sensor*, and 4) *VideoSurveillance* service for monitoring the *Video Cameras*. In addition, the server needs to handle the scheduling and routing of the generated Ethernet packets over the communication backbone. The scheduling and routing of the packets is handled by the NAC component.

Network Access Controller (NAC). It performs packet routing from server to the devices and vice versa, in addition to their scheduling. Routing is performed by a *Router* component (as shown in Figure 8 in Appendix A), which also models the network delay (assuming a given network speed of 100 Megabits per second). Scheduling is based on the type of the network packets. In our model, there are four types of packets, namely *ptp*, *event*, *audio* and *video*, with the *ptp* packets having the highest priority to be sent over the network.

Component type	Name	S	V_d	V_t	C	Size	Number
Atomic	Router	8	7	1	5-120	2^{11}	63
	Forwarder	4	1	1	5-120	2^{8}	-
	FrameReceiver	2	1	1	5-120	2^{7}	-
	MasterClock	3	1	1	0-2000	2^{12}	1
	AudioGenerator	2	1	1	0-3125	2^{13}	1
	SmokeDetector	3	1	1	0-300	2^{10}	4
	VideoGenerator	3	1	1	0-40000	2^{16}	2
Compound	NAC	-	-	-	-	2^{34}	63
	Server	-	-	-	-	2^{86}	1
	Audio Player	-	-	-	-	2^{44}	52
	Camera	-	-	-	-	2^{50}	2
	SmokeSensor	-	-	-	-	2^{51}	4
	HCS System	-	-	-	-	2^{3122}	1

Table 2: State-space estimation.

Device. Each device run one or more services which either generate packets for the server, or consumes packets generated from the server. As devices are connected in daisy chains, they also perform routing of packets, hence each device provides a NAC functionality. Services modeled in our example are *Audio Player*, PTP *SlaveClock*, *Smoke Sensor* and *Video Camera*. Video frames are generated at a rate of 25 frames per second, the size of the video frames being given as a distribution. Separate distributions are provided for high-resolution camera (with mean frame size of 120 kb) and for the low-resolution camera (with mean frame size of 30kb). The architecture of a generic device is shown in Figure 10 in Appendix A.

The table 2 gives an overview about the number and the complexity of model components defined in BIP. The columns are as follows: S is the number of control locations; V_d is the number of discrete variables (can be Boolean or arbitrary type like a frame or an array of frames); V_t is number of clocks; C is the clock range; $Size$ is the approximated size of the state-space; and $Number$ is the number of occurrences of the module in the example.

C Appendix for Section 5.1 : Parametric Precision Estimation for PTP

We introduce hereafter an analytic method to estimate the precision achieved within one round of the PTP protocol, depending on several (abstract) parameters such as the initial difference and the bounds (lower, upper) on the allowed drift of the two clocks, the bounds (lower, upper) of the communication delay between the master and the slave, etc.

The difference between the master and the slave clocks after one PTP round can be determined from a system of arithmetic non-linear constraints extracted from the model of the protocol and communication media. Let us consider one complete round of the protocol as depicted in Figure 12. The first two axes correspond to the (inaccurate) clocks of the master and slave respectively. The

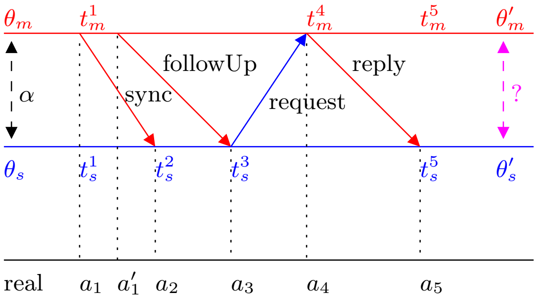


Figure 12: One round of the PTP protocol.

third axis correspond to a perfect reference clock. Using the notation defined on the figure we can establish several constraints relating initial and final values of the master and slave clocks $(\theta_m, \theta_s, \theta'_m, \theta'_s)$, timestamps (t_1, t_2, t_3, t_4), offset (o), communication delays (L_1, U_1, L_2, U_2), reference dates $(a_1, a'_1, a_2, a_2, a_4)$ as follows:

- initial constraints and initial clock difference α
  $\theta_m - \theta_s = \alpha$, $\theta_m = t_m^1$, $\theta_s = t_s^1$
- evolution of the master clock is constrained by some maximal drift ϵ_m
  $(1-\epsilon_m)(a_4 - a_1) \leq t_m^4 - t_m^1 \leq (1+\epsilon_m)(a_4 - a_1)$
  $(1-\epsilon_m)(a_5 - a_4) \leq t_m^5 - t_m^4 \leq (1+\epsilon_m)(a_5 - a_4)$
- evolution of the slave clock is constrained by some maximal drift ϵ_s
  $(1-\epsilon_s)(a_2 - a_1) \leq t_s^2 - t_s^1 \leq (1+\epsilon_s)(a_2 - a_1)$
  $(1-\epsilon_s)(a_3 - a_2) \leq t_s^3 - t_s^2 \leq (1+\epsilon_s)(a_3 - a_2)$
  $(1-\epsilon_s)(a_5 - a_3) \leq t_s^5 - t_s^3 \leq (1+\epsilon_s)(a_5 - a_3)$
- communication delays, forward (L_1, U_1) and backward (L_2, U_2)
  $L_1 \leq a_2 - a_1 \leq U_1$
  $L_1 \leq a_3 - a'_1 \leq U_1$
  $L_2 \leq a_4 - a_3 \leq U_2$
  $L_1 \leq a_5 - a_4 \leq U_1$
- internal master delay (l, u) for sending the *followUp* after *sync*
  $l \leq a'_1 - a_1 \leq u$
- offset computation and final clocks values
  $o = (t_s^2 + t_s^3 - t_m^1 - t_m^4)/2$, $\theta'_m = t_m^5$, $\theta'_s = t_s^5 - o$

This system of constraints encodes precisely the evolution of the two clocks within one round of the protocol. The synchronization achieved correspond to the difference $\theta'_m - \theta'_s$. We analyze different configurations and we obtain the following results:

1. symmetric delays $L_1 = L_2 = L$, $U_1 = U_2 = U$, no drift $\epsilon_m = \epsilon_s = 0$

$$-\frac{U-L}{2} \leq \theta'_m - \theta'_s \leq \frac{U-L}{2}$$

2. symmetric delays $L_1 = L_2 = L$, $U_1 = U_2 = U$, no master drift $\epsilon_m = 0$

$$-\frac{U-L}{2} - \frac{\epsilon_s(5U-L+u)}{2} \leq \theta'_m - \theta'_s \leq \frac{U-L}{2} + \frac{\epsilon_s(2U+2L+u)}{2}$$

3. asymmetric delays, no drift $\epsilon_m = \epsilon_s = 0$

$$-\frac{U_2-L_1}{2} \leq \theta'_m - \theta'_s \leq \frac{U_1-L_2}{2}$$

4. asymmetric delays, no master drift $\epsilon_m = 0$

$$-\frac{U_2-L_1}{2} - \frac{\epsilon_s(3U_1+2U_2-L_1+u)}{2} \leq \theta'_m - \theta'_s \leq \frac{U_1-L_2}{2} + \frac{\epsilon_s(2U_1+2L_2+u)}{2}$$

We remark that, in general, the precision achieved does not depend on the initial difference between the two clocks. Nevertheless, it is strongly impacted by the communication jitter, which is, the difference $U - L$ in the symmetric case and differences $U_2 - L_1, U_1 - L_2$ in the asymmetric case.

Moreover, we remark that in the asymmetric case, the lower and upper bounds are not *symmetric* i.e., the precision interval obtained is not centered around 0. The bounds of the interval suggest us an additional offset correction:

$$\delta_o = \frac{(U_2-U_1)+(L_2-L_1)}{4}$$

which will *shift* the interval towards 0. For example, using this additional correction we obtain in the case of asymmetric delays with no drift better precision:

$$-\frac{(U_2+U_1)-(L_1+L_2)}{4} \leq \theta'_m - \theta'_s \leq \frac{(U_1+U_2)-(L_1+L_2)}{4}$$

D Appendix for Section 5.2

In Figure 13, the solid plot shows the distribution of delays from Device(0,3) to the server and the dashed plot shows the delay from Device(3,3)to the server. One observe that the distributions differ for the two devices.

E Appendix for Section 5.3

Table 3 shows, for Device (0,0), the difference in the time and number of simulations required for PESTIMATION and SSP with the same degree of confidence.

F Graphs (larger size)

Figures 14, 15, 16, 17, 18 and 19 are large versions of Figures 4a, 4b, 5, 7a and 7b, presented in the paper.

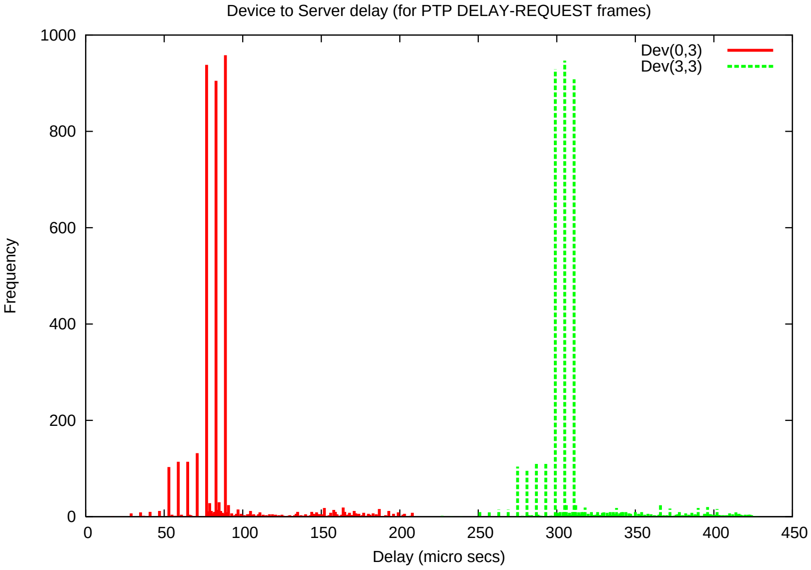


Figure 13: Delay distribution for Device(0,3) and Device(3,3).

Precision	10^{-1}		10^{-2}		10^{-3}	
Confidence	10^{-5}	10^{-10}	10^{-5}	10^{-10}	10^{-5}	10^{-10}
SSP / SPRT	110 $1s$	219 $1s$	1146 $6s$	2292 $13s$	11508 $51s$	23015 $1m44s$

Table 3: Number of simulations / Amount of time required for PESTIMATION and SSP.

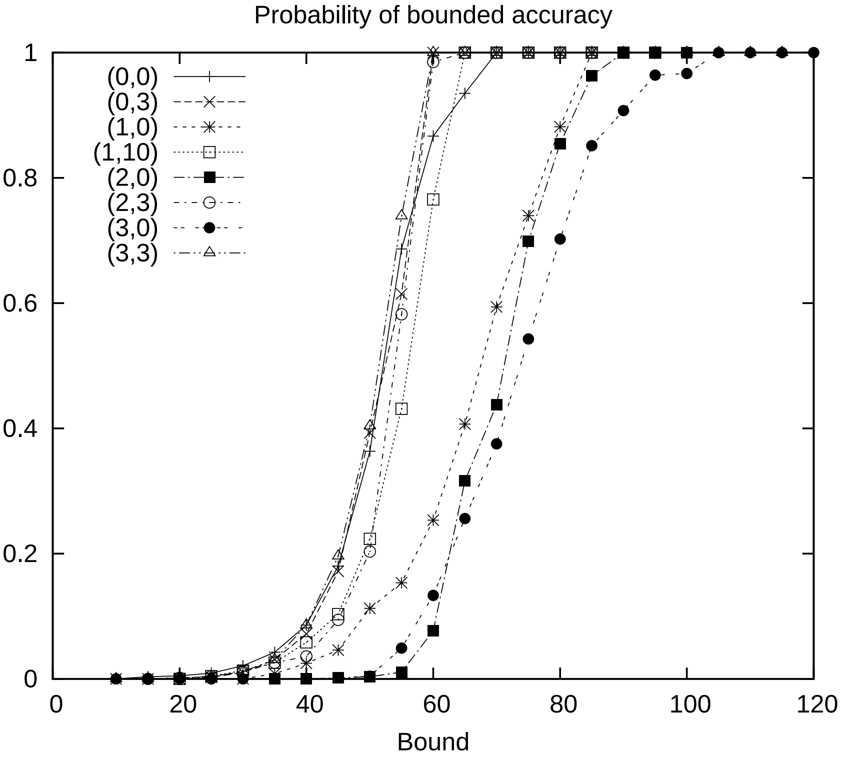


Figure 14: Probability of satisfying the bounded accuracy property as function of the bound Δ for the asymmetric version of PTP.

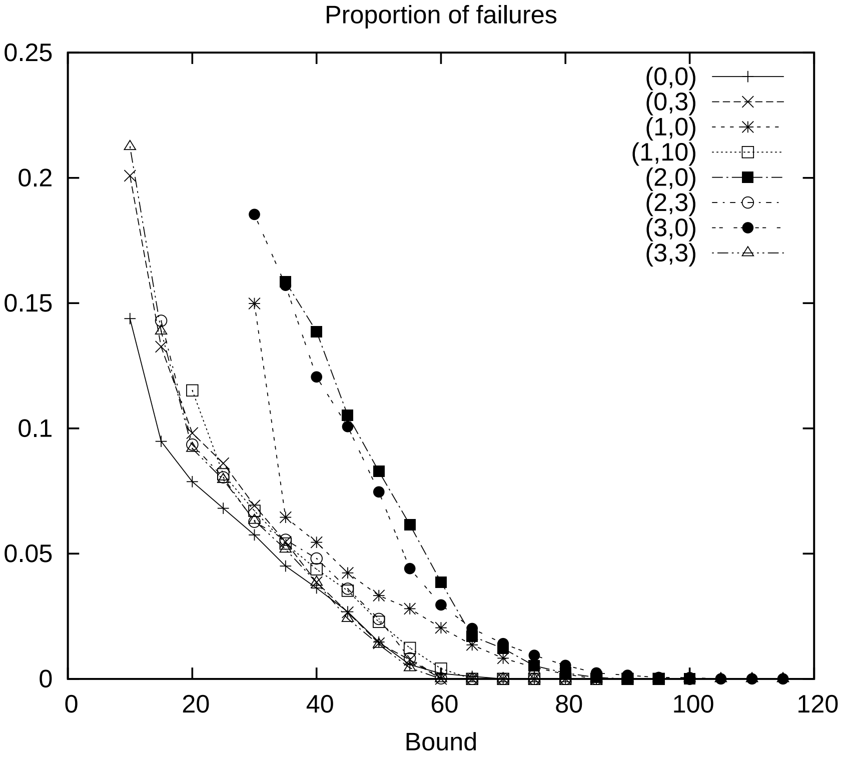


Figure 15: Average proportion of failures as function of the bound Δ for the asymmetric version of PTP.

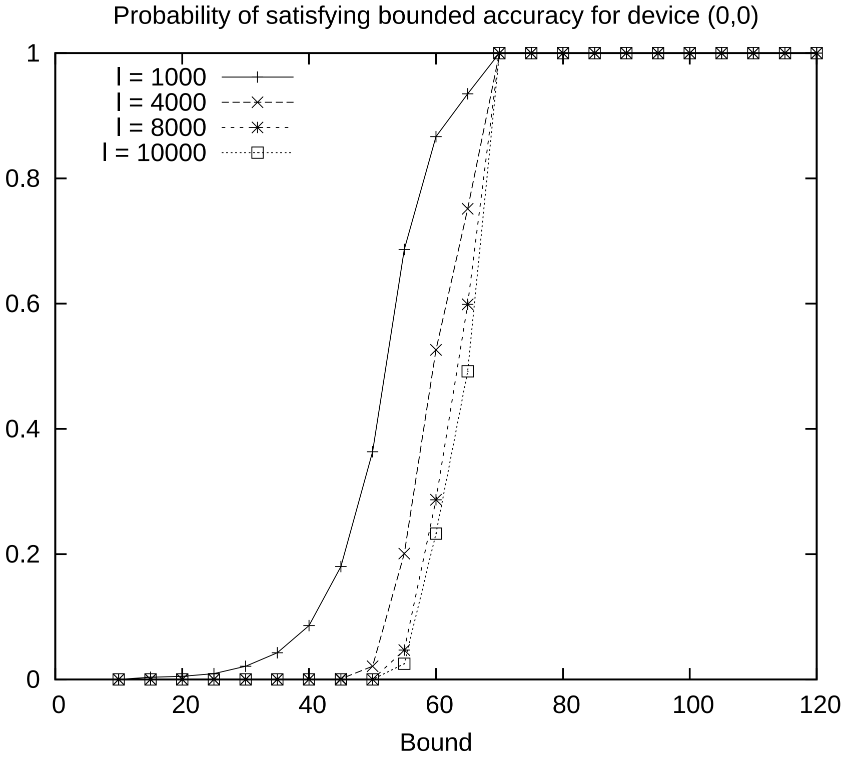


Figure 16: Evolution of the probability of satisfying the bounded accuracy property with the length of the simulations for Device $(0, 0)$ and the asymmetric version of PTP.

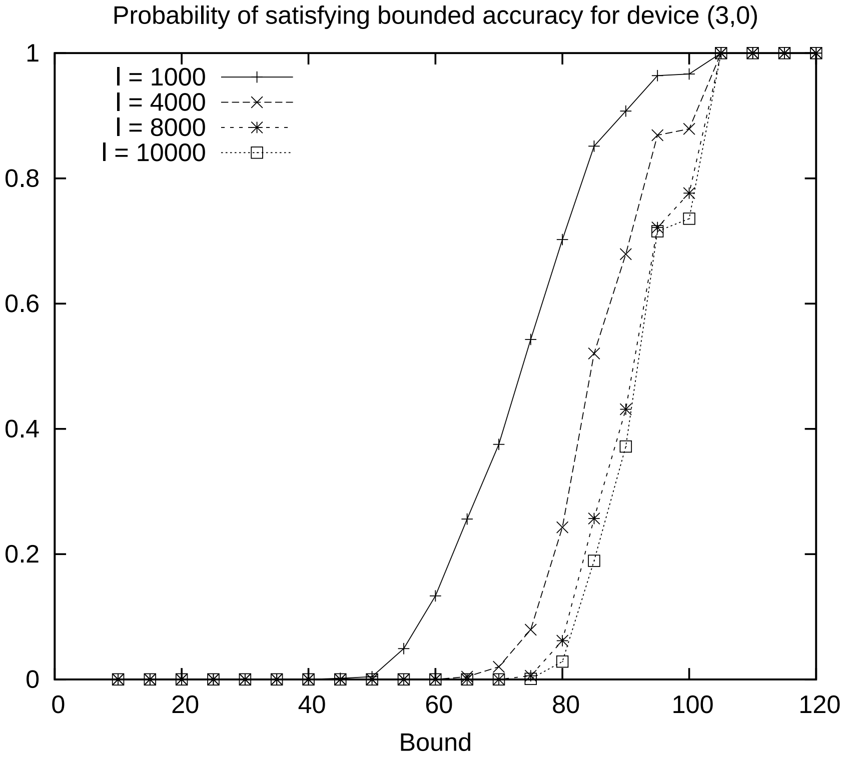


Figure 17: Evolution of the probability of satisfying the bounded accuracy property with the length of the simulations for device $(3, 0)$ and the asymmetric version of PTP.

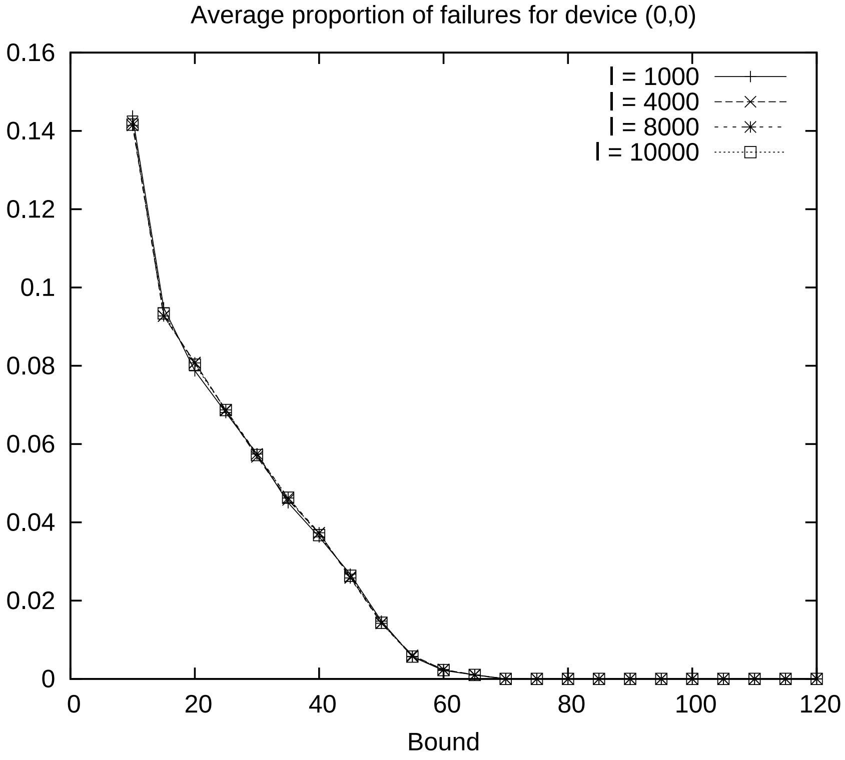


Figure 18: Evolution of the average proportion of failures with the length of the simulations for Device $(0, 0)$ and the asymmetric version of PTP.

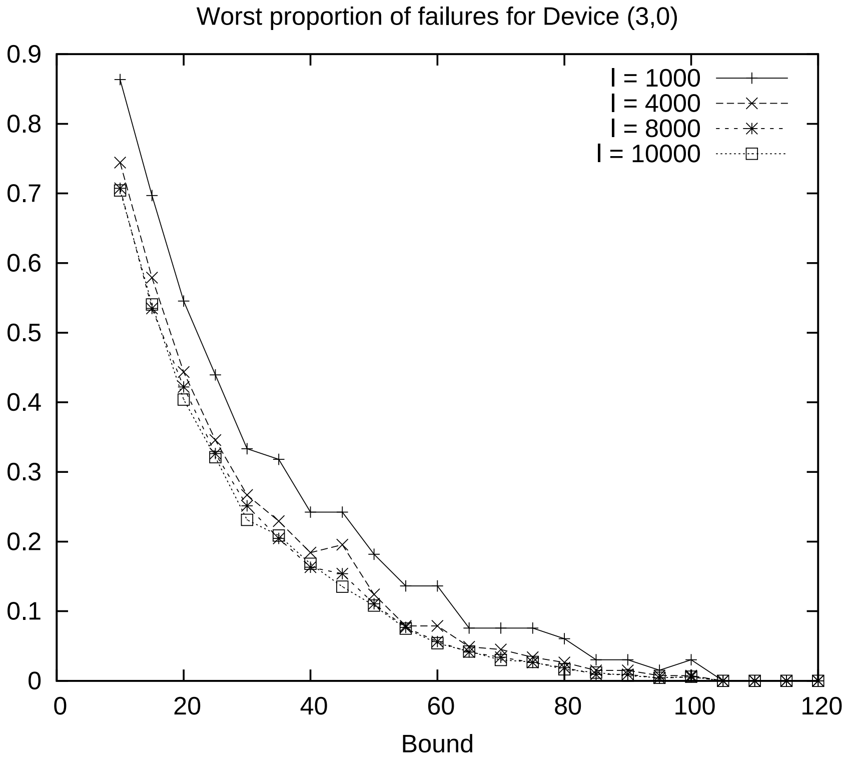


Figure 19: Evolution of the worst proportion of failures with the length of the simulations for Device $(3, 0)$ and the asymmetric version of PTP.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399