Under consideration for publication in Formal Aspects of Computing

Fun with FireWire: a comparative study of formal verification methods applied to the IEEE 1394 Root Contention Protocol

Mariëlle Stoelinga

Department of Computer Engineering
University of California at Santa Cruz, CA 95064, USA
marielle@cse.ucsc.edu

Abstract. The IEEE 1394 Root Contention Protocol is an industrial leader election algorithm for two processes in which probability, real–time and parameters play an important role. This protocol has been analysed in various case studies, using a variety of verification and analysis methods. In this paper, we survey and compare several of these case studies.

Keywords: IEEE standards, leader election algorithms, probabilistic algorithms, real-time, parameter synthesis, network protocols, applied formal methods.

1. Introduction

The IEEE 1394 Root Contention Protocol (RCP) has become a rather popular case study for investigating the feasibility of formal specification and verification techniques. The protocol is small and easy to understand and yet the problems encountered when verifying this protocol are in many aspects illustrative for the formal verification of many other applications. Moreover, as a part of the IEEE 1394 serial bus protocol ("FireWire," "iLink"), RCP is associated with an appealing state–of–the–art multimedia application.

RCP is a subprotocol of the Tree Identify Phase of the IEEE 1394 standard and its purpose is to elect a leader among two processes. Its correctness critically depends on the use of randomisation and timing delays.

This paper compares several approaches to the verification of IEEE 1394 RCP and reports on the experiences and lessons to be learned when applying formal methods to industrial systems. Notably, we survey the papers [Sha99, SV99a, BLdRT00, BST00, CS01, D'A99, KNS02, DKN02, FS01, Sto99, SV99b, SS01, HRSV02]. These case studies can be divided into three classes: papers that study the *functional behaviour*

Correspondence and offprint requests to: Mariëlle Stoelinga of the protocol (including timing and probabilistic behaviour) [Sha99, SV99a, Sto99, SV99b, SS01], case studies that employ parametric model checking to discover the *parameter constraints* needed for protocol correctness [BLdRT00, BST00, CS01, HRSV02], and studies that focus on the *performance analysis* of RCP [D'A99, KNS02, DKN02, FS01]. The results of these case studies are summarised in Table 1 on page 8.

Organisation of the paper This paper is organised as follows. We first discuss several aspects that are important in the modeling and analysis of the protocol in Section 2. Then Sections 3, 4 and 5 survey the case studies that respectively consider the functional behaviour of RCP, the parametric aspects of RCP and the performance of RCP. Within each subsection, the works are discussed in chronological order. Finally, Section 6 presents some concluding remarks. We refer the reader to Table 1 on page 8 for an overview of results.

2. Aspects in the Modeling and Analysis of RCP

For a description of RCP, the reader is referred to the introductory chapter [MRS02] of this special issue or to any of the other papers discussed below. In this section, we describe various features that play a role in its modeling and analysis.

Due to the use of random bits, the protocol is *probabilistic* in nature. *Real–time* is needed to model and analyse the root contend wait times and communication delays in the cables. Furthermore, *nondeterminism* is essential to model the fact that the timing delays (i.e. the root contend wait times and the communication delay) do not have not fixed values, but lie within intervals. Moreover, within more abstract protocol descriptions, nondeterminism models the phenomenon that, if two nodes pick the same random bits, then either one of them is elected as leader or root contention reoccurs. Finally, *parametric* models abstract from the concrete values in the protocol. Parameters of RCP are the minimal and maximal values of RC_FAST and RC_SLOW, the maximal communication delay in the cables and the probability p that the coin flip yields heads (i.e. random bit zero is drawn, in which case fast timing is selected). Note that the (maximal) communication delay is determined by two other parameters, namely the maximal cable length and the (maximal) signal propagation delay per meter. In the sequel, we deal with the delay parameter, not with the cable length and delay per meter.

Several properties are of interest for the protocol's correctness. *Safety properties* are properties stating that "nothing bad ever happens" during execution of the protocol. A crucial safety property for RCP is that at most one leader is elected. *Liveness properties* state that "eventually, something good happens." An important liveness property is that eventually at least one leader is elected (with probability one). Furthermore, *performance properties* concern the quantitative behaviour of the protocol, for instance the probability that a leader is elected within a certain amount of time or the average number of rounds needed to elect a leader. All of these properties can either be tackled parametrically or nonparametrically.

Obviously, it is impossible to consider all those features and properties at the same time. Therefore, each of the case studies described below focuses on one or more aspects, while abstracting from others.

Notation Throughout this paper, we use the following notation. Let PN denote the parent notify signal (parent request), CN the child notify signal (acknowledgement) and IDLE the idle signal. Furthermore, *delay* denotes the maximal communication delay in the channel, *rc_fast_min* denotes the minimal value of RC_FAST and *rc_fast_max* its maximal value. Similarly, *rc_slow_min* and *rc_slow_max* respectively denote the minimum and maximum of RC_SLOW. The actual values of these constants can be found in [MRS02].

3. Functional and Timing Behaviour of RCP

3.1. Verification of RCP using stepwise refinement

The papers [Sto99, SV99b, SS01] verify RCP using automata theory. That is, they describe both the protocol and its specification as automata, respectively `Impl` and `Spec`. Correctness of `Impl` with respect to `Spec` is then expressed by `Impl` $\sqsubseteq$ `Spec`, where $\sqsubseteq$ is a suitable behavioral inclusion. This relation is established by *stepwise abstraction*: it is shown that `Impl` $\sqsubseteq \mathtt{I}_1 \sqsubseteq \mathtt{I}_2 \sqsubseteq \mathtt{I}_3 \sqsubseteq$ `Spec`. The mentioned works consider different versions of `Impl` and $\mathtt{I}_1$: The automaton $\mathtt{I}_1$ is obtained by abstracting from the communication in `Impl`,

$\mathtt{I}_2$ removes all timing information from $\mathtt{I}_1$ (in the discrete time case $\mathtt{I}_1 = \mathtt{I}_2$) and $\mathtt{I}_3$, which is similar in all works, further contracts the internal choices. To prove these inclusions, [Sto99, SV99b] introduce special cases of the probabilistic simulation relations from [SL95, SV99b]. The inclusion $\mathtt{I}_2 \sqsubseteq \mathtt{I}_3$ is the step where the main probabilistic analysis is carried out and only involves small automata.

Discrete time model [Sto99] As a starting point for further verification, [Sto99] describes a probabilistic, discrete time model in the probabilistic I/O automata framework from [Seg95]. The abstraction to discrete time is justified by the observation that RC_SLOW is about 2 times RC_FAST and that the communication delay is negligible compared to the root contention wait times.

The work [Sto99] studies the probabilistic behaviour in combination with fairness and uses a combination of manual verification and model checking to establish the protocol correctness: the relations $\sqsubseteq$ are checked manually and the invariants and fairness properties needed to establish these inclusions are checked with SMV [McM93]. An important conclusion from this verification effort is that is quite easy to model the protocol in SMV and check the desired properties, but the formal relationship between the I/O automaton model and the derived SMV model, which is needed to infer that the properties that were checked for the SMV model also hold for the I/O automaton model, involves many technical details.

Real–time model [SV99b] To study the real–time behaviour, timing has been modeled more precisely in [SV99b], using the probabilistic timed I/O automaton from [Seg95]. As in the discrete time model, the communication between the nodes is modeled as the transfer of packages (PN or CN). That is, single messages which are sent only once and, upon receipt, removed from the wire. The analysis of this protocol model has been done manually, where the constants *rc_fast_min*, *rc_fast_max*, *rc_slow_min*, *rc_slow_max* and *delay* are treated as timing parameters. Two constraints on these parameters are derived that ensure protocol correctness:

$$\begin{aligned} \mathit{delay} &< \mathit{rc\_fast\_min}, && (\mathit{Eq_0}) \\ 2 * \mathit{delay} &< \mathit{rc\_slow\_min} - \mathit{rc\_fast\_max}. && (\mathit{Eq_2}) \end{aligned}$$

However, a document from the IEEE 1394 working group [LaF97] (found by the authors after publication of their work) provides different timing constraints:

$$\begin{aligned} 2 * \mathit{delay} &< \mathit{rc\_fast\_min}, && (\mathit{Eq_1}) \\ 2 * \mathit{delay} &< \mathit{rc\_slow\_min} - \mathit{rc\_fast\_max}, && (\mathit{Eq_2}) \end{aligned}$$

showing that the model in [SV99b] does not conform to the IEEE standard. Neither of these constraints are present in the standards, but the root contend wait times for the 1394 and 1394a standards meet all of them.

Detailed model [SS01] A close inspection of the IEEE documentation yielded that it is inappropriate to model the communication between the nodes by a packet mechanism as in [Sto99, SV99b]. This is for two reasons. First, it is necessary to model the absence of a message (IDLE) explicitly. Secondly, signals, unlike packages, may remain unseen by the receiving node. The latter is the case if a second signal (possibly IDLE) arrives at the receiving node's port, while the node has not sampled its port since the first signal arrived. This observation yielded the more accurate model in [SS01], where the processes communicate via signals, that is, where discrete events represent changes in the signals — the signals themselves are driven continuously across the wire.

There is one (minor) point where the model in [SS01] does not conform to the IEEE standard: initially, both nodes in this model detect root contention simultaneously, whereas one can infer from the standard that any delay less than *delay* is allowed between the detection of root contention by both nodes. When root contention reoccurs, the model in [SS01] does, however, allow for exactly this maximal delay between both detections.

Since the probabilistic analysis of this model is very similar to the real–time model, [SS01] replaces the probabilistic choice by nondeterminism and focuses the timing parameters. The work established experimentally that the equations $\mathit{Eq_1}$ and $\mathit{Eq_2}$ from [LaF97] are necessary and sufficient for correct protocol operation. To do so, it used the model checker Uppaal to analyse a large number of protocol instances with different parameters values. Although not completely formal, these experiments provide good evidence that $\mathit{Eq_1}$ and $\mathit{Eq_2}$ are indeed the required constraints. As it is the case with SMV, it was not difficult to model and analyse the protocol in Uppaal, but the formal relationship between the I/O automaton model and the Uppaal model involves many technical details.

3.2. Modeling RCP with E–LOTOS

E–LOTOS Shankland et al. [Sha99, SV99a] present a formal description in E–LOTOS of the entire Tree Identify Phase in IEEE 1394, including RCP. E–LOTOS is an enhancement to the ISO standard formal description technique LOTOS and extends LOTOS with time, a better modularity and more flexible data types. An advantage of E–LOTOS is its similarity with programming languages, making it easy to read for engineers, see [MS00]. The main purpose of this work is to investigate the usability of the new language features of E–LOTOS; the experience is positive. Since tools for E–LOTOS have not been developed yet, no rigorous verification was carried out.

Although created independently, the models [Sha99, SV99a] (their RCP parts) and [SV99b] are quite similar and both do not completely comply to the standard. Each of these works models the communication by a packet mechanism. Secondly, in [Sha99, SV99a], a CN is sent immediately after a PN has been detected, whereas the standard requires to wait at least the minimal root contention time. It is said in [SV99a, MS00] that this has been done because checking for a message after the waiting time has been expired is not expressible in E–LOTOS. If this is indeed the case, then this would plead for an extension of E-LOTOS with more expressive means.

Since no probabilistic choice is present in E–LOTOS, it is replaced by nondeterminism, as in [SS01, CS01, HRSV02]. Furthermore, being integrated in the Tree Identify Phase, the nodes in [Sha99, SV99a] automatically detect root contention asynchronously, in less than *delay* time one after another, also in the initial state. The latter is not the case in [SS01].

4. Parametric Model Checking of RCP

Given a parameterised system model $\mathcal{A}$, a property ϕ and an optional initial parameter constraint C_0, the aim of *parametric model checking* is to synthesise a parameter constraint C_1 which describes the exact conditions on the parameter values required for $\mathcal{A}$ to satisfy ϕ, where we may assume that the parameters meet C_0. Formally, C_1 is such that $C_0 \implies (\mathcal{A} \models \phi \iff C_1)$.

Although it is shown in [AHV93] that this problem is undecidable for linear parametric timed automata, several parametric model checkers for these automata been developed, namely HyTech [HHW97], LPMC [LTA98], TReX [ABS01] and a parametric extension to Uppaal [HRSV02]. Given the result in [AHV93], termination of these tools is not guaranteed, but all of them have been successfully applied to RCP. Another practical problem is that the generated constraint C_1 often comes out of the tool as a very long expression, which can usually be simplified. Currently, the simplification has to be done by hand, because none of the tools has a simplification procedure.

Besides generating the constraints for a property ϕ, parametric model checking can also be used to establish that a given constraint C is sufficient for ϕ to hold in $\mathcal{A}$. In that case, one provides C as an initial constraint checks that the generated constraint C_1 is `true`. Necessity of C can be established by checking that $\neg C$ is sufficient for $\neg\phi$. Although undecidable as well, checking parameter constraints turns out to be considerably more efficient in practice than generating them.

Since automatic parameter analysis is challenging enough, all the models below replace the probabilistic choice that governs the selection of random bits by a nondeterministic one.

LPMC Toetenel and his team ([BLdRT00, BST00]) have used their parametric model checker LPMC to investigate the timing constraints of RCP. The model in [BLdRT00] is based on to the one in [SV99b] and [BST00] is similar to [SS01]; the same timing constraints are found. However, by designating a different initial state, the model in [BST00] allows the nodes to detect root contention asynchronously when the protocol starts. Thus, the model in [BST00] is – according to the author's current knowledge of the protocol – the only one that completely conforms to the standard. Furthermore, [BLdRT00, BST00] carry out the entire verification with LPMC. This is unlike [SS01, HRSV02, CS01], where additional machinery is needed to deal with liveness and probabilistic choice. Since LPMC is capable of analysing liveness properties, [BLdRT00, BST00] simply replaced the probabilistic choice with a fairness property. This is appropriate since only functional behaviour is considered: the fairness property is implied by the probabilistic behaviour of the protocol.

Several safety and liveness properties are analysed; the most important ones being that eventually, one leader is elected and that never two leaders are elected. However, only one or two the values are taken as parameters: *delay* and (in some cases) *rc_slow_min* − *rc_fast_max*; the other values are constants. Since the time and memory consumption in the verification was very modest, one might wonder why they did not analyse a fully parametric model.

Parametric Uppaal The work [HRSV02] verified the models `Impl` in [SV99b] and [SS01] with a parametric extension of the model checker Uppaal, where all the five timing constants of RCP are treated as parameters. It establishes the constraints needed for the inclusion $\mathtt{Impl} \sqsubseteq \mathtt{I}_1$ and for the property that never two leaders are elected. This was done both by parameter synthesis and by parameter checking. Parameter synthesis is considerably more resource consuming than parameter checking. Furthermore, [HRSV02] develops and applies a technique that, when checking sufficiency of a constraint, allows certain parameters to be eliminated, while the results obtained hold for the fully parameterised model.

When checking the parameter constraints, [HRSV02] gains serious speed ups by developing (and applying) a parameter elimination technique. Due to the special format of the RCP models, where each parameter in RCP constitutes either a lower bound or an upper bound on the timing delays, certain parameters can be eliminated, while the results hold for the fully parametric model.

The special format of the automata modeling RCP allowed certain parameters to be eliminated and still to obtain general results, which gained serious speed ups.

TReX Another parametric verification of RCP has been carried out by Collomb–Annichini & Sighireanu [CS01]. Using their TReX tool [ABS01] as well as HyTech, [CS01] analysed several variations of `Impl`, including a model that allows for asynchronous detection of root contention in the initial state. It generated the constraints for the fact that at most one leader is elected, the fact that root contention is resolved if both nodes pick different random bits; and it checks the constraints for the inclusion $\mathtt{Impl} \sqsubseteq \mathtt{I}_1$. All 5 timing constants are taken as parameters and the same constraints as in [SS01, HRSV02] are established.

TReX synthesises the constraints for RCP automatically. Since TReX overapproximates the constraints for which a property ϕ does not hold (i.e. finds a necessary constraint for ϕ), several runs of the tool with different initial constraints are needed to derive the exact constraints which needed for ϕ to hold. In this way, [CS01] derives the constraints for several properties.

Furthermore, [CS01] establishes that $\mathtt{Impl} \sqsubseteq \mathtt{I}_1$ if the parameters meet the constraints *Eq₁* and *Eq₂*. Here, *Eq₁* and *Eq₂* are given as initial constraints, not synthesised. The ability of TREX to detect synchronisation failures allows it to smoothly check whether $\mathtt{Impl} \sqsubseteq \mathtt{I}_1$, whereas [HRSV02] needs rather complicated constructions on timed automata.

The performance of TReX is worse, both in time and in space, than the parametric extension to Uppaal, also in the cases where constraints were checked and not synthesised. This is explained by the fact that TReX is more general than Parametric Uppaal. In particular, it is able to deal with nonlinear constraints, which require more complex comparison algorithms. Compared with HyTech, TReX is slower, but HyTech was not able to synthesise the constraints for RCP and ran out of memory on several crucial properties, which TReX could handle successfully.

Comparison All tools are able to analyse the constraints for RCP and report the same results. Each of the approaches has its own weak and strong points – basically determined by the power of underlying tools: LPMC is the only tool that can handle fairness, but analyses only 2 parameters; parametric Uppaal is fast, TReX needs multiple runs of the tool to synthesise the constraints, but does not need complicated constructions on automata to check inclusion of behaviour. Thus, one can expect that a combination of the techniques implemented in the various tools will yield further improvements.

5. Performance Analysis of RCP

Performance analysis aims at a quantitative evaluation of a system. Interesting performance measures for RCP are the minimal probability to elect a leader within a certain time and the maximal average number of rounds needed before a leader is elected. Traditionally, performance analysis considers purely probabilistic systems, that is, systems with discrete and/or continuous probabilistic choices, but without nondeterminism. If nondeterminism is present – such as in RCP – then most performance measures are no longer expressible as a single number, but yield an interval, since the performance of the system may depend on the resolution of the nondeterminism. Thus, we can compute the *minimal*[1] probability for RCP to elect a leader in 5 rounds, but not *the* or *the average* probability to choose a leader in 5 rounds.

Several performance analysis methods have been extended for systems with nondeterminism, for instance [LSS94, BA95, Alf97, McI99, KNSS01], where only [KNSS01] has been implemented. The papers [BA95, Alf97, McI99] consider finite systems that associate to each state a number expressing the cost of being in that state. Since its real-time behaviour makes RCP an infinite state system these works cannot be applied directly here. In fact, only [LSS94, KNSS01] are directly applicable.

Hence, the approach taken by [D'A99] and [FS01] is to first remove the nondeterminism, respectively replacing it by a probabilistic and a deterministic choice, and then to apply techniques from traditional performance analysis. The works [KNS02] and [DKN02], which apply techniques from [KNSS01], are one of the few that take up the challenge of doing performance analysis in the presence of nondeterminism.

Spades D'Argenio [D'A99] investigates the performance of RCP using a discrete event simulator for ♤ (Spades). ♤ is a stochastic process algebra which allows one to specify timing delays governed by arbitrary probability distributions. Discrete event simulation is similar to testing in the sense that many runs from a system are taken, for which the performance measures are then computed. However, since all choices are probabilistic, one can exactly quantify the accuracy of the simulation — which is high if very many runs are taken.

The protocol model is based on [SV99b]. Although the standard specifies timing delays to be taken nondeterministically within their respective intervals, [D'A99] assumes a uniform distribution for the root contention times and β–distribution for the communication delay. Since techniques and tools for doing performance analysis in the presence of nondeterminism and real–time hardly existed when [D'A99] was written, resolving the nondeterministic choices by probabilistic ones was the best one could do. The analysis shows that, in most of the cases, root contention is resolved in one round of the protocol. It also reveals that both the average time until root contention is resolved and its variance grow approximately linearly with the cable length.

Deadline properties Kwiatkowska, Norman and Sproston [KNS02] and Daws, Kwiatkowska and Norman [DKN02] study *deadline properties* of RCP. Given a deadline of d ns, they compute the minimal probability that RCP elects a leader within that deadline, for different values of d. Such deadline properties can be verified automatically by the tool Prism [dAKN$^+$00, KNP02] for systems modeled as finite Markov decision processes (MDPs).

The models analysed in [KNS02] are $\mathtt{Impl}$ and $\mathtt{I}_1$ from [SS01], augmented with probabilities $\frac{1}{2}$ for the outcomes of the coin flips. Since $\mathtt{Impl}$ and $\mathtt{I}_1$ include real–time, these do not fall into the class of MDPs. Three techniques are used to translate $\mathtt{I}_1$ into a finite MDP: the first uses the forward reachability method from [KNSS01] implemented in HyTech; the second partitions the state space of $\mathtt{I}_1$ according to the region equivalence as in [AD94]; and the last one interprets $\mathtt{I}_1$ in integer semantics, which yields in this case a model that is equivalent to the standard real semantics, but which is finite. The resulting MDPs were then given as input to Prism and all yield (upon termination) the same minimal probability for electing a leader within the choose deadlines. Since $\mathtt{Impl} \sqsubseteq \mathtt{I}_1$, we know that the minimal probability to elect a leader within deadline d for $\mathtt{I}_1$ is a *lower bound* for the minimal probability for $\mathtt{Impl}$ to do so. This needs not be the exact probability, since it is not known whether $\mathtt{I}_1 \sqsubseteq \mathtt{Impl}$ – probably so, but a firm result would be useful here.

The model $\mathtt{Impl}$ was analysed using integer semantics only, which was the most efficient technique for the analysis of $\mathtt{I}_1$. Since the state space of the generated MDP grows with the deadline d, smaller deadlines can be analysed for $\mathtt{Impl}$ than for the smaller automaton $\mathtt{I}_1$. The minimal probabilities for $\mathtt{Impl}$ are the same as for $\mathtt{I}_1$, thus suggesting that $\mathtt{I}_1 \sqsubseteq \mathtt{Impl}$ indeed.

The work [DKN02] takes a similar approach and analyses the model $\mathtt{I}_1$ using Kronos and Prism. First, Kronos is used to construct the symbolic forward reachability graph, which is then fed into Prism to analyse

[1] Since these intervals can be open, half-open or closed, we should, in general, consider infima and suprema. It is not difficult to show that for the performance measures of RCP mentioned below the minima and maxima exist.

the deadline properties. The analysis with Kronos and Prism is considerably more efficient in time and space than the analysis with HyTech and Prism; the same results are found. Furthermore, [DKN02] studies the influence of using a biased coin. As mentioned in [Sto01], a curious property of the protocol is that the both the maximal and the minimal probability to elect a leader before a deadline can be slightly increased by increasing the probability for a node to select fast timing. This is because, although the protocol will need more rounds to elect a leader, the time per round in lower when both nodes select fast timing. It turns out that, for deadlines of 6000 and 10000 ns, the optimal maximal probability is reached when the probability to select fast timing is approximately 0.6.

pGCL A third investigation of the performance of RCP has been carried out by Fidge & Shankland [FS01] using pGCL (probabilistic guarded command language). The language pGCL [MM99] is a probabilistic extension of Dijkstra's GCL where pre– and postcondition predicates no longer yield booleans, but values in $[0,1]$ representing probabilities.

The model analysed in [FS01] is a high–level description of the protocol in which neither nondeterminism nor real-time is present. Instead, root contention always reoccurs when two nodes pick the same random bits. When computing the worst case performance, this abstraction is appropriate, but a formal proof to justify this would be valuable.

Using pGLC proof rules, [FS01] establishes that the probability to terminate in M rounds of the protocol equals $1-(p^2+(1-p)^2)^M = 1-(1-2p+2\cdot p^2)^M$, where p is the probability to select fast timing. Since each round is bounded by *rc_slow_max* (communication delays are neglected), the probability to terminate within a deadline $M \cdot \mathit{rc\_fast\_max}$ is at least $1-(1-2p+2\cdot p^2)^M$. These lower bounds are strictly smaller than the exact ones derived by [KNS02]. This can be explained by the fact that, if both nodes select fast timing, then a round is bounded by *rc_fast_max*, rather than by *rc_slow_max*. However, [FS01] present an easy to compute symbolic expression, whereas the tool [KNS02] computes exact bounds for fixed deadline.

Besides RCP, [FS01] also provides a pGCL analysis of the Tree Identify Phase.

Future work Several interesting performance aspects of RCP remain to be investigated, such as the minimal and maximal average number of rounds and the minimal and maximal average time before a leader is elected. The works [D'A99, KNS02, FS01, DKN02] provide a good starting point here.

A useful fact for efficiency reasons, which has been successfully exploited in [KNS02], is that the implementation relation $\sqsubseteq$ mentioned above preserves many relevant performance measures (namely those that can be expressed by traces). In other words, if $\mathcal{A} \sqsubseteq \mathcal{B}$ then $\mathcal{A}$ does not perform worse than $\mathcal{B}$ with respect to those measures. If we also have $\mathcal{B} \sqsubseteq \mathcal{A}$, then $\mathcal{A}$ and $\mathcal{B}$ satisfy *exactly* the same performance measures.

Moreover, we remark that the minimal and maximal average *number of rounds* can be computed easily with the methods by [BA95, Alf97] on the automaton $\mathtt{I}_3$, because we can abstract from the exact timing delays. However, no techniques exist yet for calculation of bounds on the average *time* before a leader is elected, so an extension of the results by [BA95, Alf97] to probabilistic timed automata would be useful here.

6. Conclusion

From the papers [Sha99, SV99a, BLdRT00, BST00, CS01, D'A99, KNS02, FS01] and our own experiences with verifying the IEEE 1394 Root Contention protocol, we conclude the following.

The verification of RCP shows again that constructing a realistic protocol model is far from easy: Industrial standards are often informal, incomplete and difficult to read for nonexperts. Thus, errors in a protocol model arise easily due to misinterpretations and misunderstandings. Moreover, it is usually unavoidable to abstract from certain details in the standard, but it is hard to judge whether these abstractions are appropriate. In RCP, it turned out to be inappropriate to model the communication delay between the nodes by a packet mechanism. Since other case studies verified parts of the Tree Identify Phase using a packet mechanism as well, it would be worthwhile to investigate to what extent this is appropriate there.

Furthermore, for maximal profit from tool support, it would be desirable to have more translations available between different formalisms and input languages of tools. If one wants to analyse a model specified in a certain formalism, with a tool having a different input formalism, then the first formalism has to be encoded into the second. These encodings, though not very difficult, involve a lot of technical details due to different languages having different synchronisation principles, fairness assumptions, priorities, etc.

paper	formalism	aspects	properties	technique	tool
[Sto99]	PIOA	dt,pr,nd	beh. incl.	sim.rel, mc	manual, SMV
[SV99b]	PTIOA	rt,pr,nd	beh. incl.	sim.rel	manual
[SS01]	TA	rt,nd,pm	beh. incl.	sim.rel, mc	Uppaal
[Sha99, SV99a]	E-LOTOS	rt,nd	–	–	–
[BLdRT00, BST00]	LPTA	rt,pm,nd,f	safety, liveness	pmc	LPMC
[CS01]	LPTA	rt,pm,nd	safety, beh. incl.	pmc	TReX, HyTech
[HRSV02]	LPTA	rt,pm,nd	safety, beh. incl	pmc	Pa-Uppaal
[D'A99]	♤	rt,pr	perf. prop.	DE	DES
[KNS02]	PTA	rt,pr,nd	deadline prop.	prmc	HyTech + Prism
[DKN02]	PTA	rt,pr,nd	deadline prop.	prmc	Kronos + Prism
[FS01]	pGCL	dt,pr	perf. prop.	WPE calculus	manual

Fig. 1. Overview of methods. Abbreviations: *formalism* PIOA = probabilistic I/O automata, PTIOA = probabilistic timed I/O automata, LPTA = linear parametric timed automata, PTA = probabilistic timed automata, *aspects* pr = probability, nd = nondeterminacy, pm = parameters, f = fairness, *properties* beh. incl = behavioural inclusion ($\sqsubseteq$), perf. prop. = performance properties, deadline prop. = deadline properties, *technique* mc = model checking, pmc = parametric model checking, prmc = probabilistic model checking, DE = discrete event simulation, WPE calculus = weakest pre-expectation calculus, *tools* Pa-Uppaal = parametric extension of Uppaal, DES = discrete event simulator for Spades.

Automating the translations between formalisms could be achieved easily and would save a considerable amount of verification effort.

Finally, it would be interesting to know why the IEEE standard has chosen this particular leader election algorithm, rather than the more obvious method: each node flips a coin and sends the outcome to the other party, until two different outcomes arise. The latter procedure seems to be faster and, importantly, does not depend on timing constraints. This is relevant as the timing constraints Eq_1 and Eq_2 require the the contention times to increase when longer cables are used to connect the nodes (see also [LaF97]). Hence, the current values of RC_FAST and RC_SLOW limit the type of applications for the 1394 serial bus. Exactly for this reason, alternative root contention protocols are investigated by the IEEE 1394 working groups ([LaF97]).

References

[ABS01] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: a tool for reachability analysis of complex systems. In G. Berry, H. Comom, and A. Finkel, editors, *Proceedings of the 13th International Conference on Computer Aided Verification,* Paris, France, volume 2102 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[AD94] R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AHV93] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. *25th Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 592–601, 1993.

[Alf97] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.

[BA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.

[BLdRT00] G. Bandini, R.L. Lutje Spelberg, R.C.H. de Rooij, and W.J. Toetenel. Application of parametric model checking – the root contention protocol using LPMC. In *Proceedings of the 7th ASCI Conference,* Beekbergen, The Netherlands, Febuary 2000, pages 73–85, 2000.

[BST00] G. Bandini, R.L. Lutje Spelberg, and W.J. Toetenel. Parametric verification of the IEEE 1394a root contention protocol using LPMC. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications (RTCSA 2000),* Cheja Island, South Korea, December 2000, pages 207–214, 2000.

[CS01] A. Collomb–Annichini and M. Sighireanu. Parameterized reachability analysis of the IEEE 1394 Root Contention Protocol using TReX. In P. Pettersson and S. Yovine, editors, *Proceedings of the Workshop on Real-Time Tools (RT-TOOLS'2001)*, 2001.

[D'A99] P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems.* PhD thesis, University of Twente, November 1999. Available via `http://wwwhome.cs.utwente.nl/~dargenio`.

[dAKN$^+$00] L. de Alfaro, M. Z. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic processes using MTBDDs and the Kronecker representation. In S. Graf and M. Schwarzbach, editors, *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems,* Berlin, Germany, volume 1785 of *Lecture Notes in Computer Science.* Springer-Verlag, 2000.

[DKN02] C. Daws, M. Z. Kwiatkowska, and G. Norman. Automatic verification of IEEE 1394 Root Contention Protocol with KRONOS and PRISM. In *Proceedings of 7rd International Workshop on Formal Methods on for Industrial Critical Systems (FMICS'02)*, 2002.

[FS01] C. Fidge and C. Shankland. But what if I don't want to wait forever? In J.M.T. Romijn S. Maharaj, C. Shankland, editor, *Proceedings of the International Workshop on Application of Formal Methods to the IEEE1394 Standard* Berlin, March 2001, 2001. submitted.

[HHW97] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. In *Software Tools for Technology Transfer*, volume 1, 1997. Also available via `http://www-cad.eecs.berkeley.edu/ ~tah/HyTech/`.

[HRSV02] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 2002. A shorter version appeared in the proceedings of *TACAS*, March 2001.

[KNP02] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *Computer Performance Evaluation / TOOLS*, pages 200–204, 2002.

[KNS02] M. Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. In S. Maharaj, C. Shankland, and J.M.T. Romijn, editors, *Formal Aspects of Computing*, 2002.

[KNSS01] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 268, 2001.

[LaF97] D. LaFollette. SubPHY Root Contention, Overhead transparencies, August 1997. Available through URL `ftp://-gatekeeper.dec.com/pub/standards/io/1394/ P1394a/Documents/97-043r0.pdf`.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing*, pages 314–323, Los Angeles, CA, 1994.

[LTA98] R.F. Lutje Spelberg, W.J. Toetenel, and M. Ammerlaan. Partition refinement in real-time model checking. In A.P. Ravn and H. Rischel, editors, *Proceedings of the Fifth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, Lyngby, Denmark, volume 1486 of *Lecture Notes in Computer Science*, pages 143–157. Springer-Verlag, 1998.

[McI99] A.K. McIver. Quantitative program logic and performance. In J.-P. Katoen, editor, *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, 1999.

[McM93] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem.* Kluwer Academic Publishers, 1993.

[MM99] C.C. Morgan and A.K. McIver. pGCL: formal reasoning for random algorithms. In *South African Computer Journal*, volume 22, pages 14–27, 1999. Special issue on the 1998 Winter School on Formal and Applied Computer Science.

[MRS02] S. Maharaj, J. M. T. Romijn, and C. Shankland. Introduction to the IEEE1394 FireWire. *Formal Aspects of Computing*, 2002.

[MS00] S. Maharaj and C. Shankland. A survey of formal methods applied to leader election in IEEE 1394. *Journal of Universal Computer Science*, pages 1145–1163, 2000.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[Sha99] C. Shankland. Using E-LOTOS to pick a leader. *Proceedings of the Workshop on Formal Methods Computation,* Ullapool, UK, September 1999, pages 143–162, 1999.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SS01] D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the IEEE 1394a Root Contention Protocol using Uppaal2k. *Springer International Journal of Software Tools for Technology Transfer (STTT)*, 3(4):469–485, 2001.

[Sto99] M.I.A. Stoelinga. Gambling for leadership: Verification of Root Contention in IEEE 1394. Technical Report CSI-R9904, Computing Science Institute, University of Nijmegen, 1999.

[Sto01] M.I.A. Stoelinga. Fun with FireWire: Experiences with verifying the IEEE1394 Root Contention Protocol. In J.M.T. Romijn S. Maharaj, C. Shankland, editor, *Proceedings of the International Workshop on Application of Formal Methods to the IEEE1394 Standard* Berlin, March 2001, pages 35–38, 2001. Also, Technical Rapport CSI-R0107, Computing Science Institute, University of Nijmegen, March 2001.

[SV99a] C. Shankland and A. Verdejo. Time, E-LOTOS and the FireWire. In M. Ajmone Marsan, J. Quemada, T. Robles, and M. Silva, editors, *Proceedings of the Workshop on Formal Methods and Telecommunications, (WFMT'99)* Zaragoza, Spain, September 1999, pages 103–119. Univ. of Zaragoza Press, 1999.

[SV99b] M.I.A. Stoelinga and F.W. Vaandrager. Root Contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)* Bamberg, Germany, May 1999, volume 1601 of *Lecture Notes in Computer Science*, pages 53–75. Springer-Verlag, 1999. Also, Technical Rapport CSI-R9905, Computing Science Institute, University of Nijmegen, May 1999.