Fun with FireWire: Experiences with Verifying the IEEE 1394 Root Contention Protocol

Mariëlle Stoelinga *
Computing Science Institute, University of Nijmegen,
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
marielle@cs.kun.nl

1 Introduction

The IEEE 1394 Root Contention Protocol (RCP) has become a quite popular case study used to investigate the feasibility of a formal verification technique. Being part of the IEEE 1394 serial bus protocol, which has been developed for interconnecting multimedia equipment (and which is also known under the names of FireWire and iLink), RCP is associated with an appealing state–of–the–art multimedia application.

RCP is an industrial leader election protocol for two processes in which both timing and probabilistic aspects are crucial. It is small, easy to understand, and yet, the problems encountered in verification of this protocol are in many aspects illustrative for the application of formal methods to other real–life applications.

Several case studies, using different tools and techniques, have analysed various aspects of the protocol. This paper compares several approaches to the verification of IEEE 1395 RCP and reports on the experiences and lessons to be learned when applying formal methods to industrial applications.

Rather than presenting new technical results, this paper aims at giving an overview of the papers [11, 12, 1, 2, 3] and previous work [14, 15, 13, 5] by the author together with Thomas Hune, Judi Romijn, David Simons and Frits Vaandrager.

2 Root Contention within IEEE 1394

The IEEE 1394-1995 standard [6] and its improvement [7] specify a high performance serial bus, suited for cheap and fast data transfer between computer and multimedia devices. The standard is described in a layered, OSI style and RCP is part of the Tree Identify Phase (TIP), present in the physical layer of the protocol.

An IEEE 1394 network consists of several nodes (devices), having one or more ports. Via their ports, nodes can be connected in a tree-like network topology. The purpose of TIP is to construct a spanning tree over this network, where the root of this tree will act as bus master in subsequent phases of the protocol.

As a basic operation, each node can drive a $PARENT\_NOTIFY$ (PN) or a $CHILD\_NOTIFY$ (CN) signal to a neighbor node, or the node can leave the line undriven ($IDLE$). The PN signal is to ask the other node to become parent (connecting closer to the root) of the sending node (then connecting further away from the root) and is acknowledged by a CN signal. The receipt of a CN signal on a port, is acknowledged by removing the PN signal from the connecting cable. In the final stage of TIP, two neighboring nodes may each try to find their parent by sending a PN signal to each other. This situation is called root contention in which RCP is initiated to elect one of the two nodes as root.

2.1 The Root Contention Protocol

If a node receives a PN signal on a port, while sending a PN signal on that port, it knows it is in root contention. Note that root contention is detected by each of the two contending nodes individually. Upon detection of root contention, a node backs off by removing the PN signal, leaving the line in the state $IDLE$. At the same time, it starts a timer and picks a random bit. If the random bit is one, the node will wait for a time RC_SLOW, whereas if the random bit is zero, it will wait for a shorter time RC_FAST. The table below lists the wait times as specified in the IEEE 1394 and 1394a standards [6, 7]. Another relevant constant is the cable velocity, which is minimally 5.05 ns/m. Since the cable length is at most 4.5 m, this yields a maximum

*PROGRESS Project TES4199, Verification of Hard and Softly Timed Systems (HaaST).

propagation delay (*delay*) of 22.8 ns.

When its timer expires, a node samples its contention port once again. If it sees *IDLE*, it starts sending *PN* anew and waits for *CN* signal as an acknowledgment. If, on the other hand, a node samples a *PN* on its port, then it sends the *CN* signal back as an acknowledgement and becomes the root. In the case that both nodes pick identical random bits, there is a chance of root contention again: each node may see an *IDLE* signal when its timer expires and both start sending the *PN* signal. In this case, both nodes detect renewed root contention and the whole process is repeated until one of them becomes root. Eventually (with probability one), both nodes will pick different random bits, in which case root contention certainly is resolved.

3 Models and Techniques

The RCP has been analysed using several models and verification techniques. This section compares the results and experiences from various studies of RCP. Section 3.1 describes the results of the verification activities the author has been involved in [14, 15, 13, 5] and Section 3.2 presents several other approaches to the analysis of RCP [11, 12, 1, 2, 3].

3.1 Models of RCP based on I/O automata

The papers [14, 15, 13, 5], all follow an automaton–based approach to verification. The protocol and its specification are both described as automata, respectively `Impl` and `Spec`, and correctness is expressed by $\texttt{Impl} \sqsubseteq \texttt{Spec}$. Here, $\sqsubseteq$ is a suitable notion of trace inclusion. The protocol correctness is established by stepwise abstraction: it is shown that $\texttt{Impl} \sqsubseteq \texttt{I}_1 \sqsubseteq \texttt{I}_2 \sqsubseteq \texttt{I}_3 \sqsubseteq \texttt{Spec}$. Here, $\texttt{I}_1$ is an automaton obtained by abstracting from the communication in `Impl`, $\texttt{I}_2$ removes all timing information from $\texttt{I}_1$ (in the discrete time case $\texttt{I}_1 = \texttt{I}_2$) and in $\texttt{I}_3$ internal choices are further contracted. The main probabilistic analysis is carried out in the step $\texttt{I}_2 \sqsubseteq \texttt{I}_3$. Since these automata are very small and they are identical for all different versions of `Impl` (see below), the method of stepwise abstraction simplifies the verification process significantly.

Discrete time model [14] As a starting point for further verification, [14] describes a discrete time probabilistic model of the protocol in the probabilistic I/O automata framework developed by Segala [10]. 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.

In this model, the probabilistic behaviour (in combination with fairness) has been studied. Most of the verification has been done manually, but several invariants and fairness properties have been have been checked with the model checker SMV [9]. It turned out that it is not so difficult to model the protocol in SMV, but the formal relation between the I/O automaton model and the derived SMV model involves many technical details.

Real–time model [15] In order to study the timing behaviour, timing has been modeled more precisely in [14], yielding the probabilistic timed I/O automaton [10]. As in the discrete time model, the communication between the nodes is modeled as the transfer of single messages (*PN* or *CN*) that are sent only once, and upon receipt removed from the wire. The analysis of this 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 correctness:

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

A document from the IEEE 1394 working group [4] (found by the authors after publication of their work) provided different timing constraints than the ones derived in [15]:

$$\begin{aligned} 2 * delay &< rc\_fast\_min, \\ 2 * delay &< rc\_slow\_min - rc\_fast\_max, \end{aligned}$$

showing that the model in [15] is not conform the IEEE standard. These constraints are not present in the standards, but the root content wait times for the 1394 and 1394a standards do meet them.

Detailed model [13] 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 [15] for two reasons. First, it is necessary to model the absence of a message (*IDLE*) explicitly. Secondly, signals may remain unseen by the receiving node. This is the case if a second signal (possible *IDLE*) arrives at the receiving node's port while the node has not sampled its port since the first signal has arrived.

This analysis yielded a more detailed model [13], where the communication has been by signals that are continuously being driven across the wire. Since the probabilistic analysis of this protocol model is very

timing constants	minimum	1394	1394a	maximum	1394	1394a
RC_FAST	*rc_fast_min*	240 ns	760 ns	*rc_fast_max*	260 ns	850 ns
RC_SLOW	*rc_slow_min*	570 ns	1590 ns	*rc_slow_max*	600 ns	1670 ns

Root contend wait times from IEEE 1394 and 1394a

similar to the real–time model, [13] only considers the timing aspects of this detailed model. This model has been verified using the timed model checker Uppaal in [13] for a large number of instances for the parameters. This analysis yielded exactly the timing constraints from [4]. As it is the case with SMV, it is not difficult to model the protocol in Uppaal, but the formal relation between the I/O automaton model and the Uppaal model involves many nasty details.

Parametric models [5] The work [5] verified the models in [15] and [13] with a parametric extension of the model checker Uppaal, where all the five constants of RCP are treated as parameters. This analysis yielded the same timing constraints.

3.2 Other Models

E–LOTOS Independently of [15], Shankland et all. [11, 12] present a formal description of RCP in E–LOTOS – an extension of LOTOS with time – of the entire Tree Identify Phase in 1394, including RCP. An advantage of E–LOTOS is its similarity with programming languages, making it easy to read for engineers, see [8]. Since tools for this language have not been developed yet, no rigorous verification is carried out for the E–LOTOS models. The models [11, 12] (the RCP part) and [15] are similar, and do not completely comply to the standard. Each of these works models the communication is by a packet mechanism. Secondly, in [11, 12], a *CN* sent immediately after a *PN* has been detected, whereas the standard requires to wait a least the minimal root contention time. It is said in [12, 8] that this 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 new expressive means.

LPMC Toetenel and his team [1, 2] have used their parametric model checker LPMC to investigate the timing constraints of RCP, where the values for *delay* and (in some cases) *rc_slow_min* − *rc_fast_max* are taken as parameters. The other values are taken as constants. The entire verification is done with LPMC, which is unlike [13, 5, 11, 12], where additional machinery is needed to deal with liveness properties and probabilistic choice. The probabilistic choice has been replaced with a fairness property. Since only functional behaviour is considered, this is appropriate, as the fairness property is implied by the probabilistic behaviour of the protocol. The model in [1] is similar to [15] and [2] to [13] and the same timing constraints are found.

Spades D'Argenio [3] investigates the performance of the RCP using the stochastic process algebra ♤ (Spades). The protocol model is based on [15]. Although the standard specifies timing delays to be taken nondeterministically within their respective intervals, [3] 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 non–determinism hardly exist, resolving the nondeterministic choices by probabilistic ones is currently the best one can do. The analysis shows that, in most of the cases, root contention is resolved in one round of the protocol and that both the average time until root contention is resolved and its variance grow approximately linearly with the cable length.

4 Conclusion

From the papers [11, 12, 1, 2, 3] and from my own experiences with the formal verification of the IEEE 1394 Root Contention, I conclude the following.

In order for the results of a formal verification to be reliable useful for engineers, the protocol models must – of course – be realistic. Constructing a realistic protocol model is, however, not easy. It is unavoidable to abstract from certain details in the standard but it is hard to judge whether these abstractions are appropriate. this is hampered by the fact that industrial standards are often informal, incomplete and difficult to read for nonexperts.

Since it turned out to be inappropriate to model the communication delay between the nodes by a packet mechanism in RCP, it is worthwhile considering to what extent this is appropriate in the other parts of the Tree Identify Phase.

For a maximal profit from tool support, it is desirable to have more established translations between different formalisms and input languages of tools. With

automated tools for those translations, a lot of time in the verification could be saved.

Finally, it is not clear why the IEEE standard has chosen this particular leader election algorithm, rather than the most obvious one (where each node sends the outcome of its coin flip to the other node until two different outcomes are tossed), which seems to be faster and easier. This question becomes more relevant (see [4]) as the timing constraints of current implementation require the contention times to be longer if the cable length between the nodes increase.

References

[1] 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 beekbergen, the netherlands, febuary 2000. *Proceedings of the 7th ASCI Conference*, pages 73–85.

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

[3] 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`.

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

[5] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. To appear in Proceedings TACAS'2001.

[6] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. Std 1394-1995, August 1996.

[7] IEEE Computer Society. 1394a Standard for a High Performance Serial Bus (Supplement). Std 1999, 1999.

[8] 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.

[9] K.L. McMillan. The smv system, draft, February 1992. Available through URL `http://www.cs.cmu.edu/~modelcheck/smv.html`.

[10] 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.

[11] 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.

[12] 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.

[13] D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. Report CSI-R009, Computing Science Institute, University of Nijmegen, Nijmegen, 2000. Submitted.

[14] 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.

[15] 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.