Under consideration for publication in Formal Aspects of Computing

The IEEE 1394 Tree Identify Protocol: Problem Specification

Savi Maharaj[1], Judi Romijn[2] and Carron Shankland[1]

[1]Department of Computing Science and Mathematics
University of Stirling, United Kingdom
[2]Department of Computing
University of Nijmegen, The Netherlands

Keywords: IEEE standard, FireWire, comparative case study, formal methods.

Abstract. We describe the leader election protocol of the IEEE standard 1394 serial multimedia bus in general terms. This protocol makes an ideal subject for a comparative case study because it is small yet complex, with many modelling decisions possible.

1. Background

Multimedia systems and devices are an increasingly common in homes and offices. Several means of fast and reliable transportation of digitised video and audio signals between these systems and devices have been developed, including, for example, the Universal Serial Bus (USB) and the 1394 High Performance serial bus (FireWire). Formal methods have long been applied to communicationing systems such as these, especially as the behaviour becomes more complex, multi-layered and difficult to fully understand. In this volume we use part of the IEEE 1394 serial multimedia bus as a case study to compare the application of various formal methods.

Case studies have an important role to play at several stages in the development of a formal method. Comparative case studies, in which different formal methods are applied to the same specification problem, are specially illuminating because the similarities and differences between the methods are thrown into sharp relief. Such studies can also serve as benchmarks to be used in asessing new developments in formal methods. Previous comparative case studies include the steam boiler control problem [1], the RPC memory case study [5], the robot production cell [14], the Light Control requirements engineering study [4], and the invoicing case study [8].

The IEEE 1394 Multimedia Serial Bus [10] ("FireWire") is especially suitable as the subject of a comparative case study for several reasons. It is an international standard, therefore there is a clear and widely available statement of its definition. The 1394 High Performance serial bus description was published as an IEEE standard in 1995 [10], and amended in the Supplement of 2000 [11]. These documents use a combination of informal English text, state transition diagrams, and C code to describe the behaviour of the bus. In

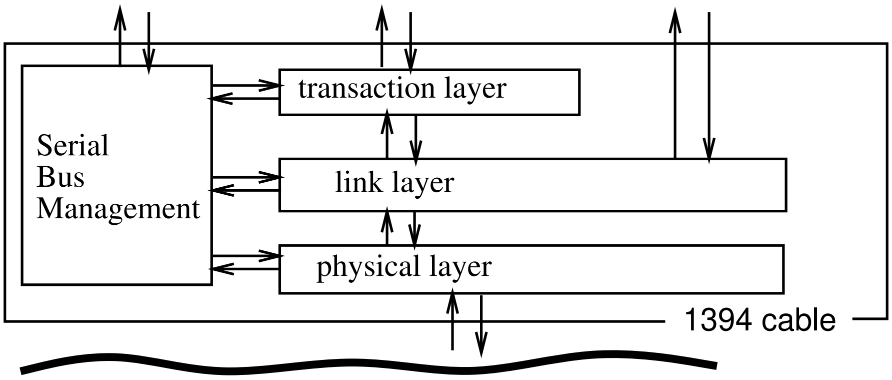

Fig. 1. Overall Architecture of IEEE 1394

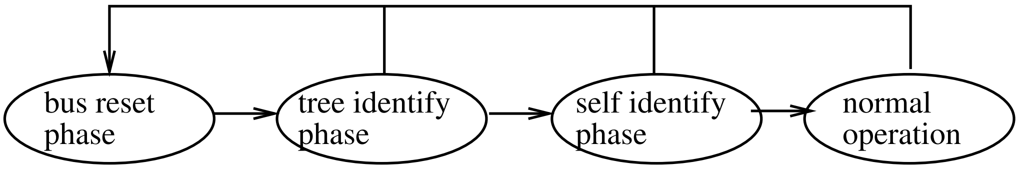

Fig. 2. Phases in PHY layer of IEEE 1394

addition, it has already been described by a number of other authors [2, 3, 6, 7, 9, 17, 18, 19, 20, 21] using formal methods, some of which are surveyed in an earlier work [16].

There is widespread interest in the correctness of the IEEE 1394 standard: companies such as Sun Microsystems, Apple, Philips, Microsoft, Sony and others have been involved in its development. Further developments of the bus include the draft 1394b for higher speed communications [12] and the draft 1394.1 for connectivity of a greater number of nodes via bus bridges [13].

Finally, the IEEE 1394 is a complex system with a variety of aspects which pose a challenge to the capabilities of formal description techniques. Formal methods can be applied in a number of ways to this problem. Firstly, they can be used to enforce precise and complete descriptions of the system (specification). Secondly, they can be used to establish correctness criteria and to detect errors (verification). Thirdly, tools may be used to check the proof or to traverse possible behaviours of the system (automation).

2. 1394 Overview

The 1394 bus is designed to connect together a number of distributed components (multimedia systems and devices). To provide maximum usability, the bus must be scalable and cope easily with changing network configurations. This means that components may be added to or removed from the network at any time.

To render the complexity of the system more accessible, the behaviour of the bus is described in layers in the style of OSI, with *physical*, *link*, and *transaction* layers, as illustrated in Figure 1.

Here we are concerned with behaviour of the physical layer (referred to in the standard as PHY). The behaviour of each layer is in turn split into different phases, each associated with a different task. For example, Figure 2 shows the PHY layer consists of the phases Bus Reset, Tree Identify, Self Identify, and Normal Operation. The Tree Identify phase is a leader election protocol, and is the focus of all of the studies in this volume.

3. Tree Identify Phase

As can be seen in Figure 2 the Tree Identify Phase takes place after a bus reset in the network. A bus reset occurs when the network configuration changes, for example, when the digital camera is connected to

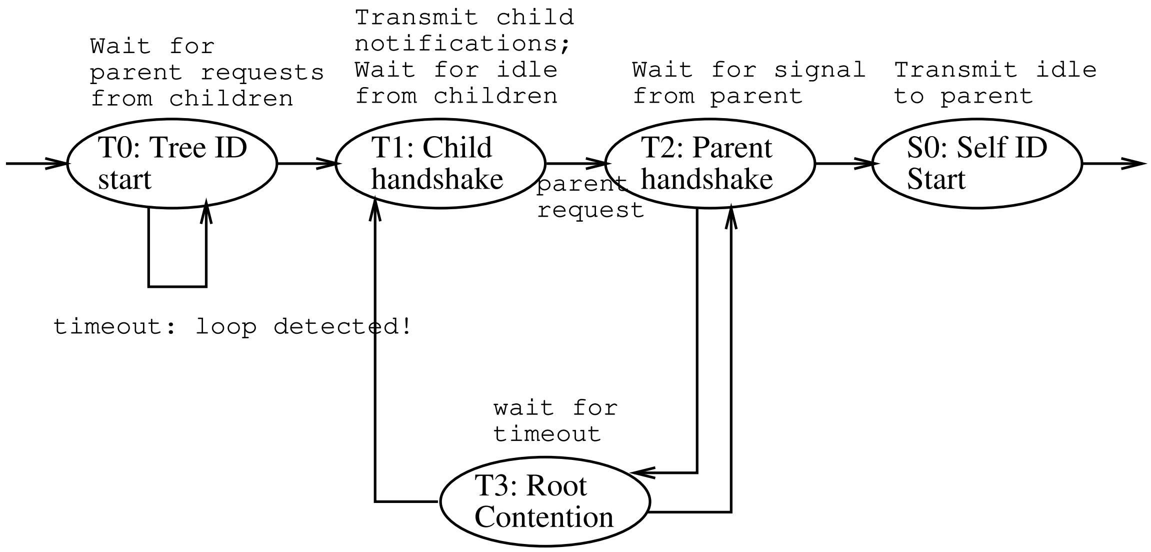

Fig. 3. Phases of the Tree Identify Protocol

the computer, or when the speakers are unplugged. The idea behind the protocol is that since the network configuration is not fixed it makes no sense to expect a particular component to always be present and be the bus manager. Therefore, a new bus manager is elected every time the network configuration changes. In the IEEE 1394 any of the components may be the bus manager.

After a bus reset all nodes in the network have equal status. They have only information about themselves; in particular, they know which of their ports are connected to other nodes. The basic structure of the protocol is shown in Figure 3. We present the protocol here in the manner of the standard, using state transition diagrams to show the behaviour, and informal text to describe what happens in these states. In addition, the standard uses C code to specify activity in each state.

Each node has exactly the same programming, so the protocol operates in an unbiased fashion. The protocol is designed for use on connected networks, will correctly elect a leader if the network is acyclic, and will report an error if a cycle is detected.

The protocol is made more complex by the problems of communication. If node A sends a request to node B then that request must be acknowledged to be sure the communication was successful.

In the perfect situation, a node will move through the three states *T0: Tree ID start*, *T1: Child handshake* and *T2: Parent handshake*, and pass on to the Self Identify phase. Broadly, the decision to move from one state to the next is based on the number of nodes requesting to be a child, c, and the number of neighbours, n.

If $n - c > 1$ the node waits for "be my parent" requests from its neighbours (potential children). This corresponds to state *T0: TreeID start*. As each request comes in c is updated. If $n - c \leq 1$ then the node transitions to state *T1: Child handshake* and starts sending "you are my child" acknowledgements to all neighbours requesting "be my parent". Children respond by ceasing to send their "be my parent" messages (the line goes idle).

If there is a loop in the network then the transition from *T0* to *T1* does not get enabled, and a timeout will generate an error message.

In the case that $n - c = 1$ (in state *T1* then the node has one neighbour which has not yet sent a "be my parent" request. It therefore asks to be a child of that neighbour by sending its own "be my parent" request. This corresponds to the move from state *T1* to state *T2*. The implication is that leaf nodes are first to communicate with their neighbours (since they have only one neighbour, so $n-c = 1$ is true immediately), so the spanning tree over the network is built from the leaves inwards.

In state *T2: Parent Handshake* (having send a "be my parent" request) the node waits for confirmation that the neighbour is its parent. When this signal is received the node transitions to *S0: Self ID start* in the next phase, which is not described here. It takes no further part in the leader election; it knows its children

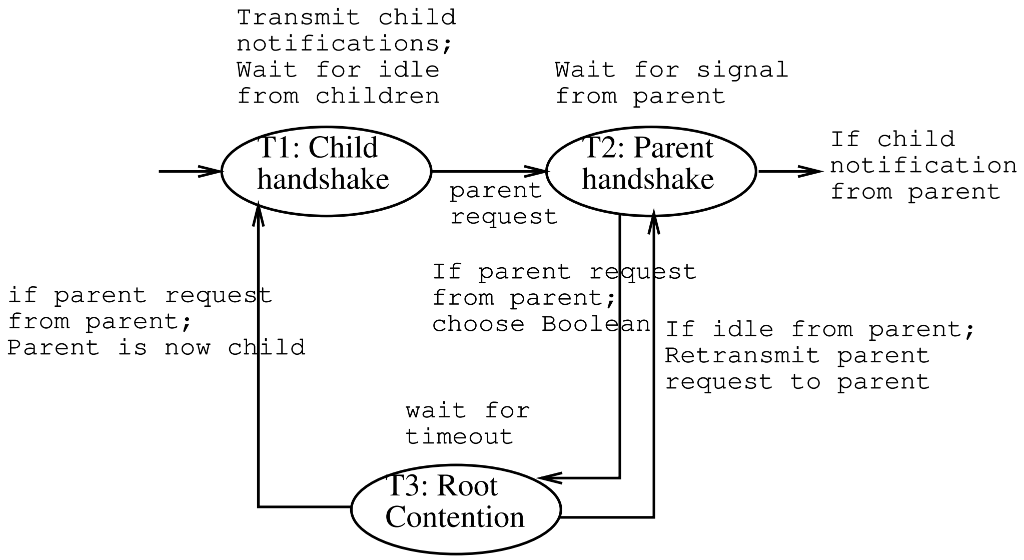

Fig. 4. Root Contention

and its parent. Problems occur if the expected confirmation is not received; this leads to *root contention*. Root contention is rather complex, and is dealt with in the next section.

3.1. Root Contention

Since requests are not atomic, *contention* may arise (two nodes simultaneously send "be my parent" requests to each other). Since only one node can be the leader contention must be resolved; this is achieved by randomisation and timing. The standard specifies that each node chooses a random Boolean and waits for a long or short time depending on the value of that Boolean. If, after the wait period is over, there is a message from the other node saying "be my parent" then this node becomes the root. If there is no such message then this node resends its own "be my parent" message (and contention may result again). This process is illustrated in detail in Figure 4.

The Tree Identify protocol is similar to one presented in Distributed Algorithms [15, page 501] where a different method of contention resolution is used (the node with the highest identifier becomes the root). This approach will not work for IEEE 1394 because the nodes do not have identifiers during the Tree Identification Phase. Identifiers are assigned in Self Identify phase; as can be seen in Figure 2, this comes after Tree Identify. The protocols were developed independently.

3.2. Terminology of the Tree Identify Protocol

Above we have used rather informal descriptions of the protocol and its messages. Messages are defined in [10, Tables 4-27 and 4-28] as combinations of voltages on wires.

IDLE	indicates a gap
TX_PARENT_NOTIFY	transmit "be my parent"
TX_CHILD_NOTIFY	transmit "you are my child"

Corresponding *received* messages are prefixed with RX instead of TX.

The standard [10, Table 4-32] and the amendment [11, Table 4-32] also define certain timing constants used in root contention.

name	definition	1394 values	1394a values
ROOT_CONTEND_FAST	the short time	$0.24\mu s$ – $0.26\mu s$	$0.76\mu s$ – $0.85\mu s$
ROOT_CONTEND_SLOW	the long time	$0.57\mu s$ – $0.60\mu s$	$1.59\mu s$ – $1.67\mu s$

The standard also lays down certain physical limitations, which in turn affect timing values. Cable length is defined to have a maximum value of 4.5m. Signal velocity is defined to be less than 5.05ns/meter. These two determine the maximum delay in sending a message from one node to its neighbour of 22.725ns.

4. Points to Address

Taking our lead from the helpful questions of Abrial et al [1] in the Steam Boiler case study, we specifically asked authors to consider the following questions during the specification process:

- Which parts of the system are specified, at which level, and in which way (verbally, rigorously, formally)? Do you consider your chosen approach to be particularly expressive or particularly awkward for this sort of problem?
- Which parts of the system are analysed, and in what way (manual, informal, partially or fully automated)?
- Have assumptions about the architecture or the behaviour of the system been made explicit and are they documented?
- Is the solution easy to change? Following a change, can analyses be reused, or are all proofs invalidated?
- How long did it take:
  - How much time has been spent on producing the solution? Give the number of person months, if possible, for the various parts of the solution.
  - How much preparation is needed to become sufficiently expert in the used specification framework in order to be able to produce a solution to such a problem in that framework? Indicate the number of weeks of training which is believed necessary for an average engineer to learn the methods.
- What are the premises for a good understanding of the proposed solution?
  - Is a detailed knowledge of the used formalism needed?
  - Can an average engineer understand the solution (with no special training)?
  - How much time do you believe is necessary for the average engineer without knowledge of the used specification methods to learn what is needed to be able to understand the solution? Could you discuss this solution with a non formal methods literate person, e.g. a customer.
  - Do you believe your solution provides a suitable alternative formalisation to the IEEE standard?

Specifically, for the Tree Identify Protocol the safety and liveness requirements to be proved are:

1. A leader is eventually chosen.
2. Only one leader is chosen.

For after the workshop:

- What are the comparable other solutions and what are the major difference with respect to them? What other solutions complement the given solutions and in which respect?

5. Results

In the final version of this introductory paper a section will be added giving an overview of how the chosen papers answer the questions posed above.

References

[1] J-R. Abrial, E. Börger, and H. Langmaack. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control.* Number 1165 in Lecture Notes in Computer Science. Springer-Verlag, 1996.

[2] 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*, pages 73–85, February 2000.

[3] 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)*, December 2000.

[4] E. Boerger and R. Gotzhein. Requirements Engineering Case Study: Light Control. *Journal of Universal Computer Science*, 6(7), 2000. Special Issue of JUCS.

[5] M. Broy, S. Merz, and K. Spies. *Formal Systems Specification: The RPC-Memory Specification Case Study.* Number 1169 in Lecture Notes in Computer Science. Springer-Verlag, 1996.

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

[7] M.C.A. Devillers, W.O.D. Griffioen, J. Romijn, and F. Vaandrager. Verification of a Leader Election Protocol - Formal Methods Applied to IEEE 1394. *Formal Methods in System Design*, 16(3):307–320, 2000.

[8] M. Frappier and H. Habrias. *Software Specification Methods: An Overview Using a Case Study.* FACIT. Springer-Verlag, 2001.

[9] W.O.D. Griffioen and F. Vaandrager. Normed simulations. In *Proceedings of CAV'98*, number 1427 in Lecture Notes in Computer Science, pages 332–344, 1998.

[10] Institute of Electrical and Electronics Engineers. *IEEE Standard for a High Performance Serial Bus. Std 1394-1995*, August 1995.

[11] Institute of Electrical and Electronics Engineers. *IEEE Standard for a High Performance Serial Bus — Amendment 1. Std 1394a-2000*, June 2000.

[12] Institute of Electrical and Electronics Engineers. *Draft IEEE Standard for a High Performance Serial Bus — (Supplement) (Gigabit 1394). P1394B-2001*, March 2001.

[13] Institute of Electrical and Electronics Engineers. *Draft IEEE Standard for High Performance Serial Bus Bridges. Std 1394.1-2001*, June 2001.

[14] C. Lewerentz and T. Lindner. *Formal Development of Reactive Systems: Case Study Production Cell.* Number 891 in Lecture Notes in Computer Science. Springer-Verlag, 1995.

[15] N. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc, 1996.

[16] S. Maharaj and C. Shankland. A Survey of Formal Methods applied to IEEE 1394. *Journal of Universal Computer Science*, 6(11):1145–1163, 2000.

[17] J.M.T. Romijn. A Timed Verification of the IEEE 1394 Leader Election Protocol. *Formal Methods in System Design*, 19(2):165–194, September 2001. Special Issue on the Fourth International Workshop on Formal Methods for Industrial Critical Systems.

[18] C. Shankland and M. van der Zwaag. The Tree Identify Protocol of IEEE 1394 in μCRL. *Formal Aspects of Computing*, 10:509–531, 1998.

[19] C. Shankland and A. Verdejo. A case study in abstraction using E-LOTOS and the FireWire. *Computer Networks*, 2001. To appear.

[20] 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*, 2001. Accepted for publication. Also available as report CSI-R009 of Computing Science Institute, University of Nijmegen.

[21] M.I.A. Stoelinga and F.W. Vaandrager. Root Contention in IEEE 1394. In *5th AMAST Workshop on Real-Time and Probabilistic Systems*, LNCS 1601, pages 53–74. Springer-Verlag, 1999.