But what if I don't want to wait forever?

[Extended Abstract]

Colin Fidge
Software Verification Research Centre
Australia

Carron Shankland
University of Stirling
United Kingdom

1 Introduction

Formal methods allow us to reason about and predict dynamic system properties. Here we show how the *probabilistic Guarded Command Language* (*pGCL*) can be used to determine the likelihood of an algorithm terminating with a successful outcome within a given deadline, where the algorithm relies on a random choice.

We focus on a specific component of the IEEE 1394 'FireWire' standard [2, 3]: *contention resolution* in the *network leader election protocol*. The leader election protocol requires a set of asynchronous communicating nodes to elect a unique leader. In certain worst cases, when the timing parameters and computation speeds of all network nodes are identical, successful leader election relies on resolution of contention (in which two nodes have nominated the other as leader). At the core of the contention resolution algorithm is a choice based on a randomly generated Boolean. This choice may result in re-entering the contentious state, so it is not obvious that root contention will ever be resolved.

Exactly this problem is addressed by Stoelinga and Vaandrager [5], who prove that the root contention resolution algorithm is guaranteed to terminate successfully *eventually*. In practice, however, a time bounded solution is more useful; how *long* do we have to wait before a leader is elected? Here we formally derive a relationship between the number of attempts to resolve contention and the probability of a successful outcome. Moreover, we use the maximum time parameters of the amendment to the IEEE standard [3] to determine the relation between the probability of successful contention resolution and the elapsed time.

2 The *pGCL* Formalism

Dijkstra's *Guarded Command Language* (*GCL*) is a widely recognised notation for describing computations [1]. It provides familiar programming language-like statements for declaration (**con** and **var**), assertion ($\{\cdot\}$), assignment ($:=$), sequencing (;), choice (**if** ... **fi**) and iteration (**do** ... **od**). Semantically, a *GCL* statement S is defined by its *weakest precondition*, $\mathrm{wp}.S.R$, a predicate describing those states from which statement S will achieve a desired postcondition R.

The *probabilistic Guarded Command Language* (*pGCL*) [4] extends this by treating pre and postcondition predicates not as Boolean-valued, but as probabilities in the range $[0, 1]$. For a particular predicate P, expression '$[P]$' equals 1 if the predicate is true and 0 otherwise. Predicate implication '$\Rrightarrow$' is interpreted as the less-than-or-equal relation on the corresponding probabilities. Most significantly, the formalism introduces a probabilistic choice operator '$S\ {}_p\!\oplus\ T$'

$$
\begin{array}{ll}
\{contenders = \{i, j\}\}; & (1) \\
k := 0; & (2) \\
\textbf{do}\ k < M \wedge \#contenders > 1 \rightarrow & (3) \\
\quad (short(i) := \text{true})\ {}_p\!\oplus\ (short(i) := \text{false}); & (4) \\
\quad (short(j) := \text{true})\ {}_p\!\oplus\ (short(j) := \text{false}); & (5) \\
\quad nodes(i), nodes(j), k := & \\
\qquad (\neg short(i) \vee short(j)), (\neg short(j) \vee short(i)), k+1 & (6) \\
\textbf{od} &
\end{array}
$$

Figure 1: Model of the IEEE 1394 root contention protocol in *pGCL*.

which chooses to behave like statement S with probability p and like statement T with probability $(1 - p)$.

3 Leader Election Protocol Model

The leader election protocol of IEEE 1394 [2, 3] may occur in two phases. In the first phase, nodes negotiate the right to be leader in a deterministic way (based on the number of neighbours and known children) until either a leader is elected, or two nodes are in contention for leadership, i.e., each wants the other to be leader. A second probabilistic phase resolves contention between the two remaining candidates by using an algorithm based on a random choice. [The full paper addresses both phases but in this abstract we consider only the second.]

Figure 1 presents our *pGCL* model of the protocol's probabilistic phase. We take a high level view of the protocol, describing properties of the network as a whole rather than describing the parallel behaviour of individual nodes. Each node has *contender* status, represented by a Boolean, if it has the potential to become leader. The overall system state is a function, *nodes*, from node identifiers to contender status. We introduce variables i and j to identify the two remaining candidate nodes at the end of the protocol's deterministic phase, and formally state the assumption that no nodes other than i and j are contenders (line 1). [The full paper proves that the deterministic phase of the protocol establishes this condition.] We define *contenders* to be the set of all nodes with contender status 'true' and $\#contenders$ to be the number of remaining contenders.

In the standard [2, 3], contention between nodes i and j is resolved as follows. When contention is detected node i chooses a random Boolean and waits for either a 'short' or 'long' time, depending on that Boolean. After waiting its chosen time, node i checks for a 'TX_PARENT_NOTIFY' message from node j, meaning that node j would like node i to be the leader. If no such message has arrived then node i resends its own 'TX_PARENT_NOTIFY' message, i.e., node i wants node j to be the leader. Node j behaves similarly. The possible outcomes of an attempt to resolve contention are therefore either that (a) both nodes wait a short time and yield to the other, (b) one node waits a short time and yields to its opponent (which is waiting a long time), or (c) both nodes wait a long time, and yield to each other. In the first and third cases, contention is not resolved and the procedure must be repeated. Note that outcome (b) is twice as likely as either (a) or (c) since there are two ways to achieve it (either

i waits a short time and j waits a long time, or vice versa).

Our model abstracts from the details of contention resolution, while maintaining its basic characteristics. It allows the two nodes to attempt to resolve contention until either a single leader is elected or a given time limit expires. The time limit is modelled by a maximum number M of attempts to elect a leader. Loop counter k keeps track of the number of attempts and is initialised to 0 (line 2). (Both k and M are auxiliary features of our model allowing us to reason about the number of attempts to resolve contention.)

An attempt to resolve contention is modelled by the decision to wait for a short or long time, and the subsequent updating of each node's status. We are not concerned at this point with the actual duration of an attempt. Function *short* indicates which nodes have decided to wait for a short period of time before offering the leadership to the other node. At each attempt to resolve contention, node i probabilistically decides whether to wait a short time or a long time (line 4). Similarly for node j (line 5). A multiple assignment then models the actions of the two nodes updating their state in response to these decisions, and the attempt counter being incremented (line 6). A given node (e.g., i) remains a candidate for leader either if it decides to wait for a long time ($\neg$ $short(i)$) or its opponent waits for a short time ($short(j)$). The loop guard tells us that the protocol terminates when either the time limit expires ($k = M$), or the number of contenders no longer exceeds one, in which case a leader has been successfully elected (line 3).

4 Formal Reasoning

Just as we can calculate a weakest precondition for a *GCL* model [1], we can use formal rules to calculate a *weakest pre-expectation* for a *pGCL* model [4]. This tells us the probability that an algorithm involving random choices will achieve a desired postcondition. Here we calculate a lower bound for the probability that a leader will be elected, given that the number of attempts is bounded by constant M. The desired postcondition is that only one contender remains.

$$R \stackrel{\text{def}}{=} [\#contenders = 1]$$

Firstly, we must determine the probability that the loop in Figure 1 will achieve postcondition R, using the conditions in the *pGCL* proof rule for probabilistic loops [4]. To do this we must determine the probability that the loop will terminate. This is trivially '1' in this case, since the loop terminates when $k \geq M$, and variable k is guaranteed to increase at every iteration.

We then define a *probabilistic invariant* [4] which represents the probability of the loop successfully terminating from an arbitrary state. Let '$\sqcup$' be the maximum operator and q denote expression '$p^2 + (1-p)^2$'.

$$\begin{aligned} I \stackrel{\text{def}}{=} \; & [contenders = \{i\} \vee contenders = \{j\}] \\ & \sqcup ((1 - q^{M-k}) * [contenders = \{i, j\}]) \end{aligned}$$

On the left of invariant I we have a predicate representing the 'successful' state where only one of the two nodes is still a contender. If this predicate is already true then the probability of success is immediately 1; we have elected a leader. Otherwise, the right-hand term is the probability of taking a 'good' step, i.e., a step resulting in a leader being elected, in one of the remaining $M - k$ iterations given that the only contenders are i and j. The complex numerical element of this invariant arises from the following reasoning: a 'bad' step is when either both nodes wait a short time, with probability p^2, or both nodes wait a long time, with probability $(1-p)^2$. Thus the probability of a bad step in the next iteration is $p^2 + (1-p)^2$, called q for convenience. Given that, the probability of a bad step in all remaining iterations is q^{M-k}, and thus the probability of a good step is any of the remaining iterations is $1 - q^{M-k}$.

Having defined our invariant, we have to show that when the loop guard (line 3) is false and invariant I holds then we have achieved the desired postcondition R.

$$[\neg(k < M \wedge \#\mathit{contenders} > 1)] * I \quad \Rrightarrow \quad [\#\mathit{contenders} = 1]$$

[This is proven in the full paper.] We must also show that the loop body preserves the invariant. This is done by showing that the weakest pre-expectation of the loop body (lines 4–6) is 'implied by' the loop guard (line 3) multiplied by invariant I.

$$[k < M \wedge \#\mathit{contenders} > 1] * I \quad \Rrightarrow \quad \text{wp}.(\text{line } 4).(\text{wp}.(\text{line } 5).(\text{wp}.(\text{line } 6).\text{I}))$$

[This is proven in the full paper.] This result, and the termination argument above, allow us to use the proof rule for probabilistic loops [4] to conclude that the probability of the loop in Figure 1 achieving postcondition R is at least as high as the probabilistic invariant I.

Finally, we can use the $pGCL$ assignment semantics [4], the loop initialisation (line 2) and assertion (line 1) to trivially put a lower bound on the probability of initialisation establishing invariant I.

$$(1 - q^M) \quad \Rrightarrow \quad \text{wp}.(\text{line } 1).(\text{wp}.(\text{line } 2).\text{I})$$

Thus, the probability of the entire algorithm successfully electing a leader in M attempts is *at least* $1 - q^M$, that is, $1 - (p^2 + (1-p)^2)^M$.

We can now use this result to predict the likelihood of successful termination of the contention resolution algorithm within a given worst-case deadline, by substituting various values for M and 0.5 for p (since contention is resolved by choosing a random Boolean, and we assume the choice is unbiased). The time for each attempt to resolve contention is variable (either short or long). From the amendment to the standard [3] the wait times for nodes in contention are in the ranges 0.76μs – 0.85μs ('short', or 'ROOT_CONTEND_FAST') and 1.59μs – 1.67μs ('long', or 'ROOT_CONTEND_SLOW'). Processing and transmission times are negligible compared to these times and are therefore ignored. Here we are interested only in the worst case time, which is the maximum wait time (1.67μs).

Attempts (M)	**Deadline** ($M * 1.67\,\mu$s)	**Probability** ($1 - q^M$)
1	1.67μs	0.5
2	3.34μs	0.75
5	8.35μs	0.96875
10	16.7μs	0.99902724

We can thus conclude with high confidence that a leader will be elected in a short period, even in the worst case where all nodes have exactly the same timing parameters.

References

[1] E.W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976.

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

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

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

[5] 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.