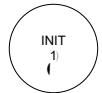


simulator or rapid prototype and such automation may increase the number of scenarios that can be examined and the reliability of the examination.

Automated examination of scenarios can be taken still further using model checking. In model checking, the case explosion the oracle or the

The node controllers and the central guardians collectively implement the Time-Triggered Protocol TTP/C that guarantees correct operation of the system despite faults in some of the

w



2.3.2 Guardian Startup

A faulty node could masquerade as another during startup, send cs-frames at inappropriate times (or continuously), and generally fail to follow the algorithm. A central guardian can mask these faults, but to do so (and to perform its prime function of enforcing the TDMA schedule

and UPPAAL . Lönn [8] considers startup algorithms for TDMA systems similar to TTA and verifies one of them using UPPAAL. However, model checking for timed automata is computationally complex, so that when we add the case/state explosion caused by considering a large number of fault scenarios, the model rapidly becomes computationally infeasible. Our initial experiments did use timed automata and we were unable to consider more than a very small number of faults.

It is essential to the utility of model checking for exploration and verification of fault-tolerant algorithms that we are able to consider a large number of different kinds of faults—ideally, we would like the

machine used for the explicit-state experiments. These two or three orders of magnitude improvement in performance encouraged us to try using model checking during development of the new startup algorithm.

However, fully exhaustive fault models still posed a challenging prospect, so we developed a modeling “dial” that could inject varying *degrees of* faults: our idea was to use as high a degree (i.e., as many kinds) of faults as proved feasible in practice. In the remainder of this section we first present our basic model of the startup algorithm and then describe the modeling concepts for faulty components of varying degrees. Due to space limitations we only give ~~examples of how to model faulty components and how to verify their correctness~~.

```

LT_TO: ARRAY index OF NATURAL = [[j : index] 2*n+j];
CS_TO: ARRAY index OF NATURAL = [[j : index] n+j];

```

The unique timeouts for each node are specified as LT_TO (listen timeout) and CS_TO (cold-start timeout), as defined in Section 2.

3.1.1 Node Model

We specify the input and output variables of an individual node as follows.

```

node[id: index]: MODULE = BEGIN INPUT
  msg_in: ARRAY channels OF msgs,
  time_in: ARRAY channels OF index,
  lock_in: ARRAY channels OF BOOLEAN
OUTPUT
  msg_out: ARRAY channels OF msgs,
  time_out: ARRAY channels OF index,
  state: states,
  counter: counts,
  errorflag: BOOLEAN

```

The _in represents the kind of message that the node receives from the hubs; if it is a normal message, then time_in indicates the slot position transmitted in the sender's frame, which equals the current time measured relative to the start of the TDMA round if the sender sends a correct value. We can

think of this inform

```
[] % Let time advance
    state = init AND counter < startupdelay
--> state' = state;
    counter' = counter+1;
    msg_out' = msg_out;
    time_out' = time_out;
```

Here, the [character introduces a set of guarded commands, which are separated by the [] symbol; the % character introduces a comment. A SAL guarded command is eligible for execution in the current state if its guard (i.e., the part before the -->) is

```
hub[c: channel s]: MODULE = BEGIN  
INPUT  
    msg_in: ARRAY index OF msgs,  
    time_in: ARRAY index OF index,  
    interlink_msg_in: msgs,  
    interlink_time_in: index
```

A hub receives msg_in and time_in as input values from each node, and interlink_msg_in and interlink_time_in from the other channel (a also listens to the other channel during startup).

```
OUTPUT  
    msg_out: ARRAY index OF msgs,  
    time_out: ARRAY index OF index,  
    interlink_msg_out: msgs,  
    interlink_time_out: index,  
    states: [0..10],  
    lock: ARRAY index OF BOOLEAN
```

A hub has the following outputs: msg_out, the kind of message the hub sends to the nodes; time_out, this variable represents

This guarded command again represents a set of transitions. The precondition is satisfied if the hub is in hub

This example of a transition by a faulty hub is activated if an attached node sends a message other than quiet to the hub. The faulty hub

-

5.1 Effectiveness of Statespace Reduction Measures

5.4 Automated Verification

nodes	feedback	eval.	cpu time	BDD
#				

References

- [1] A. Ademaj, G. Bauer, H. Sivencrona, and J. Torin. Evaluation of f