The Fault Detection Problem
(Extended Abstract)

Andreas Haeberlen
MPI-SWS
ahae@mpi-sws.org

Petr Kuznetsov
TU Berlin/Deutsche Telekom Laboratories
pkuznets@acm.org

To make a system robust against Byzantine faults, there are at least two different (and largely complementary) approaches we can use: We can add redundancy to mask the symptoms of a limited number of faults, so that users can be provided with correct service even when faulty components exist, and/or we can detect the presence of faults and identify the faulty components, so that they can be isolated and repaired. Over the last few years, great progress has been made in developing the first approach, but the second approach is still not very well understood.

Fault detection has been extensively studied in the context of “benign” crash faults, where it is assumed that a faulty component simply stops taking steps of its algorithm [1, 2]. However, very little is known about fault detection for more general classes of faults, such as Byzantine faults. There is a paper by Kihlstrom et al. [4] that discusses Byzantine fault detectors for consensus and broadcast protocols, and there are several algorithms for detecting certain types of non-crash faults, such as PeerReview [3] and SUNDR [5]. However, many open questions remain; for example, we still lack a formal characterization of the types of non-crash faults that can be detected in general, and there are no known lower bounds on the message complexity of detection.

The goal of our current work is to take a first step towards a better understanding of general fault detection. We have developed a formal model that allows us to formulate the fault detection problem for arbitrary faults, including non-crash faults, and we introduce the notion of a fault instance, which is characterized by an execution, the assigned algorithm, the set of correct nodes, and a distinct set of suspects that contains at least one faulty node. Solving the fault detection problem for a set \( F \) of fault instances means finding a transformation \( \tau_F \) that, given any algorithm \( A \), constructs an algorithm \( \bar{A} \) (called an extension of \( A \)) that works exactly like \( A \) but does some additional work to identify and expose faulty nodes. Whenever a fault instance from the set \( F \) appears, \( \bar{A} \) must expose at least one faulty suspect (completeness), it must not expose any correct nodes (accuracy), and, optionally, it may ensure that all correct nodes expose the same faulty suspects (agreement).

So far, we have used our model to answer two specific questions: Which faults can be detected, and how many additional messages does fault detection require? To answer the first question, we have shown that the set of all fault instances can be divided into four non-overlapping classes, and that the fault detection problem can be solved for exactly two of them, which we call commission faults and omission faults. Intuitively, a commission fault exists when a node sends messages a correct node would not send, whereas an omission fault exists when a node does not send messages a correct node would send.
To answer the second question, we have studied the message complexity of the fault detection problem, that is, the ratio between the number of messages sent by the most efficient extension and the number of messages sent by the original algorithm. We have derived tight lower bounds on the message complexity for commission and omission faults, with and without agreement. Our results show that a) the message complexity for omission faults is higher than that for commission faults, and that b) the message complexity is (optimally) linear in the number of nodes in the system, except when agreement is required for omission faults, in which case it is quadratic in the number of nodes.

References


