Failure Detectors in Omission Failure Environments

Danny Dolev* Roy Friedman† Idit Keidar ‡ Dahlia Malkhi§

We study failure detectors in asynchronous environments using a novel **generic** formulation of failure detection properties which generalizes several previous works. We focus on an asynchronous environment that admits message omission failures. We adapt Chandra and Toueg's definitions of failure detection Completeness and Accuracy to the omission failure model, and define an eventual weak failure detector, $\diamond\mathcal{W}(om)$, that allows any majority of the processes that become connected (and remain connected) to reach a Consensus decision despite any number of transient communication failures in their past. We provide a protocol that solves the $\lfloor\frac{n-1}{2}\rfloor$-resilient Consensus problem in this model, regardless of past omissions. Our protocol is efficient in that it requires a process to buffer and re-send only the last issued message to overcome omissions.

Generic failure detectors

We study failure detectors in asynchronous environments parameterized by the fault types that they detect, such as crash failure or Byzantine failure, and by a notion of "mutual cooperation": In every failure model, a co-operation predicate – *coop* – is defined among all pairs of processes such that $coop_p(q)$ denotes that q is "co-operating" with p. $coop_p(p)$ will usually be taken to mean that p is correct. For example, in the crash failure model $coop_p(q)$ is chosen to mean "both p and q are correct (not crashed)".

Failure detectors are defined by combination of the properties below; *Completeness* indicates success in detecting processes for which *coop* is false, and *Accuracy* indicates the ability to avoid suspecting processes for which *coop* is true:

Strong *coop* **Completeness** If $coop_p(p)$ and $\neg coop_p(q)$ then p eventually permanently suspects q.

Weak *coop* **Completeness** If $coop_p(p)$ and $\neg coop_p(q)$ then there is some process r s.t. $coop_r(p)$ and r eventually permanently suspects q.

Eventual Strong *coop* **Accuracy** If $coop_p(q)$ then there is a time after which p does not suspect q.

Eventual Weak *coop* **Accuracy** If $coop_p(p)$ then there exists some r s.t. $coop_p(r)$ and there exists a time after which for every q, s.t. $coop_q(r)$, q does not suspect r.

Note that in the definitions above p and r may be the same process.

Omission failure model

Using our generic framework we characterize the omission failure model and the failure detector classes in it.

In our model, processes may fail by crashing and in addition messages may be omitted. The members of the largest *permanently connected component* in the system[1] are considered correct. Processes outside this component are faulty. By a permanently connected component we mean a group of processes that communicate without loss and do not receive any messages from processes outside the component. Members of a **majority** connected component are called *core* processes.

We define an eventual weak failure detector for the omission failure environment, $\diamond\mathcal{W}(om)$, satisfying *Weak om Completeness* and *Eventual Weak om Accuracy*. Our definition makes use of a *majority stability* predicate *om*: $om_p(q)$ holds if p and q are core processes (i.e., belong to a permanently connected majority-component).

Weak *om* **Completeness** If there exists some core process p then for every non-core process q there exists some core process r that eventually permanently suspects q.

Eventual *om* **Weak Accuracy** If there exists some core process p then there exists some core process r such that there is a time after which no core process q suspects r.

In the full paper, we present a protocol for solving $\lfloor\frac{n-1}{2}\rfloor$-resilient Consensus (where n is the number of processes in the system) in the omission failure environment using a failure detector in $\diamond\mathcal{W}(om)$. Our protocol is practical in the sense that processes need only buffer and retransmit the last message they sent, and the only source of unboundedness in the size of messages stems from the need to carry a counter. Finally, we argue that $\diamond\mathcal{W}(om)$ is the weakest failure detector for solving $\lfloor\frac{n-1}{2}\rfloor$-resilient Consensus in the omission failure model.

*Computer Science Institute, The Hebrew University of Jerusalem.
†Department of Computer Science, Cornell University.
‡Computer Science Institute, The Hebrew University of Jerusalem.
Supported by the Israeli Ministry of Science.
§AT&T Labs–Research, Florham Park, New Jersey, USA.

[1] if there are two or more such components of equal size, **deterministically** choose one among them.

To appear in the *ACM Symp. on Prin. of Distributed Computing (PODC 97), August 1997.*