**Distributed Systems: Concepts and Design**

***Edition 3***

**By George Coulouris, Jean Dollimore and Tim Kindberg**
**Addison-Wesley, ©Pearson Education 2001**

# Chapter 11: Correctness and the Consensus Problem

Tim Kindberg, February 2003

I am indebted to Christian Worm Mortensen for an exchange about the notion of "correctness" that led to the following observations.

On p. 422, we define correctness informally thus: "Whatever the type of failure, a *correct* process is one that exhibits no failures at any point in the execution under consideration." However, there may sometimes be ambiguity in what we should consider "the execution under consideration".

Take the reduction of *RTO-multicast* to consensus on p. 455, which deals only with crash failures. Recall that the reduction proceeds by having all processes *RTO-multicast* their proposed values, and by each process choosing the first value to be reveived. Recall also that the definition of Integrity for Consensus (p. 452) is as follows:

> *Integrity*: If the correct processes all proposed the same value, then any correct process in the *decided* state has chosen that value.

Now suppose in our reduction that the process whose value was delivered first failed immediately after sending it. If that process had chosen a different value from all the correct processes that delivered its message, would that not violate the Integrity property for Consensus?

It depends on what we take to be the events in the execution under consideration. If we take the execution under consideration to end when all events at all processes that took part in this protocol have taken place, then the process that sent the chosen value was not correct for the duration of that execution and integrity was broken.

However, it can also be argued that the spirit of Integrity in Consensus should be to take Integrity as referring to correctness up to two distinct points in the execution for each individual process: the point of proposal and the point of decision. Any process that proposed a value (that is, one that *initiated* a call to *RTO-multicast*) was correct at the first point. It is the values of all such processes that should be considered in verifying Integrity, since all such values are good enough for consideration (assuming that there are no Byzantine failures). If the crashed process proposed a unique value, then the antecedent of Integrity's conditional does not apply and Integrity is not broken.

As we point out on page 452, other definitions of Integrity may also be deemed appropriate and occur in the literature. We invite the reader to consider what definition should be used in the case of some practical applications.