On Safety and Timeliness in Distributed Data Management
(Preliminary Report)

Danny Dolev
H. Raymond Strong

IBM Almaden Research Center
San Jose, CA 95120
email: dolev@ibm.com, strong@ibm.com

1 Introduction

Two basic and somewhat conflicting goals govern distributed data management - safety and timeliness. Safety means ensuring consistency always, even across network partitions. Timeliness is the guarantee that every transaction will be completed within a bounded time. One may wish to ensure that both will hold while overcoming faults. Unfortunately, as we will argue in this paper this is usually not possible. When faults can bring processes to a logical separation safety and timeliness cannot coexist.

In this paper we study three problems, and using an axiomatic approach we prove or reprove impossibility results for three basic coordination problems in distributed data management.

The atomic commit problem for distributed database transactions ([BHG, G]) has many different variants, each one tuned to optimize a different parameter. In the atomic commit problem, participants are data management processes residing at various sites and communicating by messages. When a distributed transaction involves modification of data at more than one site, the participants must coordinate whether and when to commit these changes. For reasons independent of faults and beyond the scope of the problem, any participant may unilaterally decide that the transaction should be aborted. But participants may give up their right to such a unilateral decision and guarantee that they will make the appropriate modifications if the transaction is committed. This guarantee is typically accompanied by expensive transmissions to some stable storage device and by the locking of resources that cannot be touched by other transactions until the decision to commit or abort is reached.

Presumably most of the time all participants agree to commit. The problem is to devise a protocol so that once all participants have made their unilateral decisions, the transaction is either committed or aborted by all processes, that continue to function correctly, within a bounded time, despite communication and process failures. Furthermore, when any failed process recovers it must be able to ascertain whether the modifications should be made or not as part of its recovery procedure. The well known two phase commit protocol solves most of the problem, but the failure of a single process can force others to wait indefinitely in what is called a blocking state, neither committing nor aborting the changes. Three phase commit protocols have been developed that tolerate process crash failures but cannot tolerate partition of communication between functioning processes [S,SS,DS]. We will show that it is the multivalent nature of the
outcome together with the necessity to decide, even in isolation, that makes a complete solution to the atomic commit problem impossible.

Note that this result does not contradict the results of [SS], the difference resides in the important distinction between information transfer and messages. In [SS] an “optimistic” model is explored in which communication is synchronous and a message is guaranteed delivered or returned (but not both). Understanding of the basic lower bounds we prove will lead to defining other models in which safe timely protocols exist. The model of [SS] provides a good example of messageless information transfer: both processes know when a message that was sent has not been received. In this model single partitions can be tolerated but if multiple partitions are possible, then a safe and timely commit becomes impossible.

The earliest discussed glimpse at this phenomenon and one of the most intuitive comes as the Chinese Generals’ Problem [G]. In this problem, we are given a situation with two generals of allied armies encamped on opposite hills above a valley full of the enemy. The generals’ problem is to coordinate a time of attack. However, the only way they can communicate is by messenger through the valley of the enemy. The problem is to devise a protocol for this communication that guarantees that they either both attack (at the same time) or they both retreat (at the same time). The difficulty is that the sender cannot know if a given message arrived unless an acknowledging message is returned. As part of a solution protocol, the generals may have rearranged that if no messages get through by a given time, then they will both retreat; but there must be some possibility of attack. We will show that it is the multivalent nature of the outcome together with the possibility that a decision may have to be made in isolation that makes a solution to the Chinese Generals’ Problem impossible.

One of the most famous recent impossibility results in computer science is the result of [FLP] that it is impossible to reach consensus in an asynchronous system in the presence of at most one process crash failure, even when eventual communication is guaranteed. The consensus problem is a coordination problem in the sense that all the processes that do not fail must agree on a common output. Multivalence is introduced because each process is given an input from a set of at least two values, and if all inputs agree, then all outputs must agree with the inputs. Note the similarity to the atomic commit problem: each process has some “input” that allows it to make a unilateral decision on whether to abort or allow commit; if all inputs allow commit then the agreed output must be commit. There are only two essential distinctions between the two problems. One is the constraint placed on processes recovering from failure in the atomic commit problem. The consensus problem is easier to solve because it puts no constraints on the actions taken by processes once they have failed. The other is the asynchronous environment associated with the consensus problem. A fundamental characteristic of an asynchronous environment is that external information is only conveyed to a process when it receives a message, no information can be learned from the absence of messages. We will show that asynchrony plus the possibility of process failure provides a multivalent system that is vulnerable to isolation and that neither consensus nor safe atomic commit is possible.

This negative result is easier than [FLP], because our model of a completely asynchronous system allows communication to be stopped forever. The model of [FLP] provides for guaranteed eventual communication; but allows a process to stop forever. We show how to adapt the proof of [FLP] to our paradigm. Furthermore, we show that, even in a model with guaranteed eventual communication and guaranteed eventual process repair, safety requires blocking. Our contribution here is the conversion of the [FLP] result to results in other models that fall within the scope of our theory of distributed systems. We can also take much simpler proofs of impossibility in totally asynchronous systems and translate them into results about systems with eventual process repair.

Axiom A5 below, the compatibility axiom, is the one that allows us to construct a set of