In this second part of the book, we ask how distributed computing systems can be made reliable—a question motivated by our review of servers used in Web settings, but that also tries to generalize beyond the ways that today’s cloud platforms implement reliability, so that what we learn might also apply to future web services that may be introduced by developers of new classes of critical distributed computing applications down the road. Our focus is on communication technologies, but we do review persistent (durable) storage technologies based on the transactional computing model, particularly as generalized to apply to objects in distributed environments. Our goals here will be to develop the core mechanisms needed; the third and final part of the book applies these mechanisms to Web Services and explores some related topics.

Earlier, we talked about the sense in which the cloud, as architected today, often makes a deliberate decision to weaken properties in order to guarantee snappy response. As we will see, the underlying tension involves the costs of reliability for a distributed application. To offer strong guarantees, such as the forms of consistency that Brewer’s CAP conjecture talks about, a distributed system sometimes needs to pause while it repairs damage done by a failure. For example, if it senses that a message was lost, the system might delay subsequent updates until the missing message has been recovered through retransmissions and applied to the system state in the right order. If a server crashes, the system might pause briefly to clean up anything it was doing just at that instant, for example by rolling back a partially completed update to the service state, or by completing a multicast if the crash happened just as the multicast was being sent. These events can slow responsiveness: they put the reliability property ahead of speed of response, if you want to think about the prioritization of goals that they implicitly reflect.

Even from these simple cases we see that when a system makes promises, it also accepts obligations that can involve delayed responses. In the cloud, where fast local response is key to scalability, we can turn this around by saying that the forms of reliability meaningful in cloud services will often be limited to those that can be achieved without delay. Which properties fall into this class?
It turns out that we will be able to say a great deal about such questions. Reliable distributed computing is a much-studied topic, and while the cloud brings new challenges such as guaranteeing stability even at large scale, as the saying goes, “plus ça change; plus c’est la même chose”. We will find that the new issues posed by the cloud do not change the foundational principles. Moreover, while not all forms of reliability scale well enough to use casually in the cloud, some consistency and reliability models we do not have all the answers yet, but can already be sure that any future science of reliable cloud computing will be an outgrowth from the more basic science of reliable distributed computing.