The Need for Realistic Failure Models in Protocol Design

Idit Keidar
Massachusetts Institute of Technology
Lab for Computer Science,
E-mail: `idish@theory.lcs.mit.edu`

Keith Marzullo
University of California, San Diego,
Department of Computer Science and Engineering,
E-mail: `marzullo@cs.ucsd.edu`

1 Introduction

Fault tolerant algorithms are often designed under the assumption that no more than t out of n processes or components can fail. This approach was pioneered by the SIFT project [22], and has since been widely applied to the design of algorithms for real critical systems, e.g., air traffic control [6], other highly available services like file servers [15], and so on. It is such a common assumption that most fault tolerant algorithms found in the literature today adopt it without any justification (e.g., [14, 19]).

It is a common assumption because the t out of n model gives one a simple abstraction for reasoning about failure-prone environments and system reliability. With this assumption it is fairly easy to design and verify protocols and also to express lower and upper bounds. Unfortunately, when adopting this assumption, we often forget the relationship between the t out of n assumption and system reliability.

In real systems reliability is typically expressed in terms of the probability that the system meets its specification. A more refined model expresses survivability as a range of probabilities for the system meeting a number of different degraded specifications [13]. When characterizing the possible failures as t out of n, one is implicitly expressing an upper bound on system reliability as the probability that t failures or less occur throughout the time the algorithm runs.

Forgetting the relation between the t out of n assumption and system reliability can lead to a foolish design. For example, some consensus protocols (e.g., [7]) have a structure in which once t failures have been detected the protocol proceeds with the assumption that no further failures will occur. A more sensible design would have the protocol become more cautious under these circumstances: if t failures have occurred then it is possible that the failure analysis done in computing t was faulty, and so further failures may be more likely under these circumstances.

Such foolishness can occur in a more subtle manner. In this position paper we remind ourselves of the relation between failure assumptions and system reliability. We bring to the surface some implicit assumptions made by the t out of n failure characterization and discuss their limitations. We briefly discuss failure assumptions that address some of these limitations. We argue that not all the shortcomings of this model have adequate solutions as of yet.

2 Limitations of Existing Failure Models

The t out of n characterization of failures is implicitly based on the following assumptions: (1) all failure types are equally likely; (2) the probability of a component failing while a protocol is in progress is independent of the duration of the protocol; (3) all components that can fail have an identical probability of failure; and (4) failure probabilities of different components are mutually independent. These assumptions do not adequately reflect the nature of real-world network environments, as we explain below.

Assumption (1) has *any* kind of failure counted as one against the budget of t failures. Yet, different failures can have different probabilities of occurring. This is true both for failures of different components and different failure modes of the same component. Assumption (1) is relaxed by *hybrid failure models* [20], which have separate budgets for different kinds of failures, e.g., crash, omission, and arbitrary failures. Indeed, hybrid failure models are often used in the design and analysis of fault tolerant system, e.g., [16].

Assumption (2) is only valid for protocols that run for a brief period. In practice, however, the likelihood of t failures occurring while a protocol is running is highly dependent on the protocol's duration. Thus, while consensus protocols that execute more rounds can tolerate more faults, the occurrence of more faults with such protocols is also more likely, which can lead to reduced system availability or reliability, as observed, e.g., in [1, 10].

Stochastic (probabilistic) models are often used to reflect the effect of a protocol's duration on the likelihood of failures. Such models (e.g., [1, 8]) generally assume that processes fail after an exponentially distributed random time, independently of each other. If a protocol runs in discrete steps of equal duration, then this means that there is a fixed failure probability p that a given process fails in a given step.

Such stochastic analysis is generally based on enumerating the states a protocol can be in, and modeling the probability of each state transition. In order to reduce the complexity of the model and to avoid state explosion, stochastic analyses often model several protocol steps as one atomic action. If such simplifications are not done with care, they can lead to incorrect analyses. For example, several analyses of the availability dynamic voting algorithms (e.g., [5, 11]) modeled a step where all processes agree to move into a new configuration as one atomic action. In practice, such agreement involves communication, and further failures can occur while such communication is taking place. Such failures can lead to blocking, which was not taken into account in these analyses. As shown in [10], blocking can have a major impact on system availability, especially as failures become more frequent.

Assumptions (3) and (4) are rarely questioned in protocol design. Hybrid failure models still make assumptions about the independence of failures within each category, e.g., that all processes have the same likelihood of crashing and that such failures are independent. Process failures can often be highly correlated; many spectacular stories exist of how a single fault has brought down a large collection of processes (e.g., the crash of the entire Grapevine distributed mail system and the difficulty in restarting it during 1983 SOSP [21] or the Code Red worm's ability to infect all Microsoft web servers [3]).

Similarly, protocols designed to overcome message loss and probabilistic analyses of such protocols, (e.g., [2, 8]) are usually based on the assumptions that all links are equally reliable, and that all messages are lost with equal probability and independently. In practical networks, however, this is not so. In running a group membership protocol over the Internet [12], we observed a great variability of loss rates over time. We also observed that packets sent between certain pairs of processes can be an order of magnitude more likely to be lost than packets sent between other pairs of processes. Many protocols use multicast services to send messages to multiple recipients. In such protocols, it is not uncommon that a message lost by one receiver will be lost by many [23]. Moreover, studies of the Internet multicast backbone, Mbone, have shown that in typical multicast sessions some links are much lossier than others [9] and that the mean loss rate on a given link also varies with time, sometimes abruptly and sometimes gradually [23].

3 Suggested Research Directions

Many of us who specify problems in fault tolerance and who design and analyze fault tolerant algorithms are directing our efforts towards Internet-like environments. We believe that there are three research directions that need to be pursued to enable those like us to make our work relevant.

(1) *Obtain data on how such environments fail.* This is an active research direction (e.g., [24, 4]) that is still in its infancy. Our own work on measuring the performance of the Moshe group membership service over the Internet [12] yielded many surprises, such as a non-transitive communication relation that persisted for almost a half an hour. We are currently running a long term experiment whose goal is to measure the frequency and duration of such anomalous communication relations. We are also analyzing long time scale IP-level Internet topologies to determine what the upper bounds are on the reliability one can obtain in Internet communication.

(2) *Find ways to model this data so it will be easy to reason about.* This is a difficult direction to follow. We are working on models for systems in which processes have correlated crash failures. We have come up with simple characterizations for some reasonable correlated failure assumptions, but many other failure assumptions lead (not surprisingly) to algorithms that are NP-hard.

(3) *Design protocols using more realistic failure models.* This is the most intellectually rewarding direction to follow. We are already working in this direction. For example, with Moshe we observed that with our original all-to-all protocol communication pattern (which maximized connectivity) a single lossy link could have a significant effect on performance. After studying the loss-rates between different pairs of processes we reconfigured the protocol communication pattern to avoid using particularly lossy links. In our example, communication from Taiwan to other sites was very lossy, but it was the least lossy with Cornell. We thus had Taiwan use Cornell as a portal for communication with the other sites. Doing so led to a significant improvement in performance. We are studying ways to account for dynamic changes in loss-rates by using *adaptive* strategies.

Another example is our work in [10] in which we studied the effect of frequent failures on the availability of dynamic voting algorithms. We observed that algorithms that use pipelining to halve the time needed for reconfiguration, could also halve the probability of reconfiguration being interrupted by another failure. Since failures during reconfiguration can lead to blocking, using pipelining is even more effective than one might first imagine.

A third example leverages from the observation that consecutive message losses observed by a receiver are often due to the same lossy link [17]. One can use this observation to build a caching-based loss-recovery scheme for a reliable multicast protocol [18]. One can cache the identity of the process that requested a retransmission for the previous lost message and also the identity of the responder to the last request. When a loss is observed, one can first assume that it is on the same link so the same requester and responder will be used. Doing so would speed up the recovery and eliminate duplicate requests/responses in the case of a cache hit.

Acknowledgements

Idit Keidar's research in this area is supported in part by Air Force Aerospace Research (OSR) contract F49620-00-1-0327. Keith Marzullo's research in this area is supported in part by DARPA contract N66001-01-8933 (Reliable Adaptive Multi-Path Networks). This grant is jointly held with Stefan Savage and Geoff Voelker of UCSD.

References

[1] Ö. Babaoğlu. On the reliability of consensus-based fault-tolerant distributed computing systems. *ACM Trans. Comput. Syst.*, 5(4):394–416, 1987.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, 1999.

[3] CERT. Code red worm exploiting buffer overflow in IIS indexing service DLL. CERT Advisory CA-2001-19.

[4] B. Chandra, M. Dahlin, L. Gao, and A. Nayate. End-to-end wan service availability. In *Third Usenix Symposium on Internet Technologies and Systems (USITS01)*, Mar. 2001.

[5] I. R. Chen and C. Wang, D. Analyzing dynamic voting using Petri nets. In *IEEE Symp. on Reliable Distributed Systems (SRDS)*, 1996.

[6] F. Cristian, B. Dancey, and J. Dehn. Fault-tolerance in air traffic control systems. *ACM Trans. Comput. Syst.*, 14(3):265–286, Aug. 1996.

[7] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in Byzantine agreement. *J. ACM*, 37(4):720–741, October 1990.

[8] H. S. Duggal, M. Cukier, and W. H. Sanders. Probabilistic verification of a synchronous round-based consensus protocol. In *16th IEEE Symp. on Reliable Distributed Systems (SRDS)*, Oct. 1997.

[9] M. Handley. An Examination of MBone Performance. Research Report RR-97-450, USC/ISI, Jan. 1997.

[10] K. W. Ingols and I. Keidar. Availability study of dynamic voting algorithms. In *21st International Conference on Distributed Computing Systems (ICDCS)*, pages 247–254, April 2001.

[11] S. Jajodia and D. Mutchler. Dynamic voting algorithms for maintaining the consistency of a replicated database. *ACM Trans. Database Syst.*, 15(2):230–280, 1990.

[12] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. A client-server oriented algorithm for virtually synchronous group membership in WANs. In *20th International Conference on Distributed Computing Systems (ICDCS)*, pages 356–365, April 2000. Full version: MIT Technical Memorandum MIT-LCS-TM-593a, June 1999, revised September 2000.

[13] J. Knight and K. Sullivan. Towards a definition of survivability. In *3rd Information Survivability Workshop (ISW)*, pages 85–89, Oct. 2000.

[14] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[15] E. K. Lee and C. A. Thekkath. Petal: Distributed virtual disks. In *7th International Conference on Architectural Support for Programming Languages and Operating Systems*, 1996.

[16] P. Lincoln and J. Rushby. Formal verification of an interactive consistency algorithm for the draper FTP architecture under a hybrid fault model. In *9th Annual Conference on Computer Assurance (Compass)*, pages 107–120, Gaithersburg, MD, June 1994. IEEE Washington Section.

[17] C. Livadas. Designing a caching-based reliable multicast protocol. In preparation.

[18] C. Livadas, I. Keidar, and N. Lynch. Designing a caching-based reliable multicast protocol. In *the International Conference on Dependable Systems and Networks (DSN) Fast Abstracts Supplement*, pages B44–B45, July 2001.

[19] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.

[20] F. J. Meyer and D. K. Pradhan. Consensus with dual failure modes. *IEEE Transactions on Parallel and Distributed Systems*, 2(2):214–222, April 1991.

[21] M. D. Schroeder. Personal Communication on the Grapevine correlated failure, 1983.

[22] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. M. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock. SIFT: design and analysis of a fault-tolerant computer for air traffic control. *IEEE*, 66(10):1240–1255, Oct. 1978.

[23] M. Yajnik, S. B. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *18th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE/INFOCOM'99)*, volume 1, pages 345–352, New York, NY, Mar. 1999.

[24] Y. Zhang, V. Paxson, and S. Shenkar. The stationarity of Internet path properties: Routing, loss and throughput. Technical report, AT&T Center for Internet Research at ICSI, May 2000. http://www.aciri.org/.