4th Symposium on Reliability in Distributed Software and Database Systems. Silver Spring, Maryland (Oct 1984

THE RELIABILITY OF (k,n)-RESILIENT DISTRIBUTED SYSTEMS

Y.C. Tay*

Sequoia Systems, Inc.
Boston Park West
Marlborough MA 01752

Abstract Recent efforts at achieving reliability in distributed computing include the incorporation of fault-tolerance into system software, such as routing tables, and the introduction of extra software to mask processor failures, such as Byzantine agreement. A set of n processors running software that can tolerate k faults is called a (k,n)-resilient system. Most (k,n)-resilient systems have a higher probability of failure in the long run, and a smaller mean time-to-failure, than one processor. Further, large systems fail in a very well-defined period, and they fail very quickly if n is more than linear in k.

1 Introduction

Processors have a nonzero probability of failing. Recently, there has been considerable interest in the design of distributed systems that can tolerate processor failures. This fault-tolerance is sometimes achieved by incorporating it into the software, and sometimes by adding software.

The simplest example is majority voting, where n processors perform the same computation and vote on the result. Faulty processors are outvoted by the nonfaulty ones as long as $n \geq 2k+1$, where k is the number of faulty processors. (There is also a clock synchronization protocol for $n \geq 2k+1$ in [8].)

The second example is Byzantine agreement [9], where n synchronous and unreliable processors try to agree on a value through unauthenticated messages. It is known [11] that there exists a Byzantine agreement protocol that can tolerate k failures if and only if $n \geq 3k+1$. This is true despite relaxation of the requirements to Weak [7], Crusader [2], and Approximate agreement [4]. It is also true for asynchronous agreement if authentication is allowed [1]. Byzantine agreement protocols have been used for distributing input in replicated computing [14], and for commiting transactions in distributed databases [10].

The third example is a protocol that finds routes through a network of unreliable processors. Using authenticated messages, the protocol finds a set of routes from a sender to a receiver such that p of these routes do not contain faulty nodes. Such a protocol is given in [5] for $n \geq 3k+p$.

The fourth example again concerns setting up routes through a network of unreliable processors. In the previous example, the network is changing, with old links broken and new ones added. The set of paths from a sender to a receiver must therefore change with the network. In this example, however, the problem is to construct a fault-tolerant network with a fixed routing table for a given set of processors. For each sender-receiver pair, the table gives a set of routes, so that, if the sender sends copies of a message through these routes, the receiver will get a copy as long as one of the routes is nonfaulty. A network for $n=2^{k+1}$ is given by [3].

We call the set of processors, together with the fault-tolerant software they run, the *system*. Hence, in the second example, the processors and the Byzantine agreement protocol form the system; in the fourth example, the processors and the routing table form the system. We say a system is *(k,n)-resilient* if it has n processors, and the software can tolerate at most k processor failures

* On leave from the National University of Singapore.

This work was supported by National Science Foundation Grant ECS-82-03925.

(that is, if there are more than k failures, there is no guarantee that the system will produce the right result, or continue to function). Hence, in the above example, the systems are $(k,2k+1)$, $(k,3k+1)$, $(k,3k+p)$, and $(k,2^{k+1})$-resilient respectively. Note that we do not define the notion of processor failure, since this depends on the context (as in the examples). We say a (k,n)-resilient system *fails* if the number of processor failures exceed k. Clearly, for a given n, the larger k is, the longer it takes for the system to fail. Conversely, for a given k, the larger n is, the sooner the system will fail.

This short note takes a look at the net effect on the reliability of a (k,n)-resilient system when the fault tolerance k cannot be increased without increasing the size of the system n, as in the examples.

2 Probability of failure and MTTF

We measure the reliability* of a (k,n)-resilient system by the probability that the system has failed before a given time, and by the mean time-to-failure (MTTF) of the system.

We assume that a processor has a constant failure rate λ, so that the probability density function of its time-to-failure t is the exponential distribution $\lambda e^{-\lambda t}$, where $t > 0$. This assumption has been justified both empirically and theoretically [12, 6]. We also assume that the time-to-failure of all processors are independent and identically distributed, that there are no failures at time 0, and that a failed processor remains faulty henceforth. Using only elementary probability theory, one can prove from these assumptions the following (see [12]) :

Proposition The probability that a (k,n)-resilient system will fail before time t is

$$F(t;k,n,\lambda) = \sum_{j=k+1}^{n} \binom{n}{j} (1 - e^{-\lambda t})^j \, e^{-(n-j)\lambda t}$$

$$\text{where } t \geq 0 . \qquad (1)$$

[]

* Reliability has a formal definition in the engineering literature [13]. We will not define reliability formally.

It follows [12] that the time-to-failure of a (k,n)-resilient system has probability density

$$f(t;k,n,\lambda) = \binom{n}{k} (1 - e^{-\lambda t})^k (n-k)\lambda e^{-(n-k)\lambda t}$$

$$\text{where } t \geq 0 \qquad (2)$$

and the mean time-to-failure (MTTF) is

$$\mu(k,n,\lambda) = \frac{1}{\lambda} \sum_{j=0}^{k} \frac{1}{n-j} . \qquad (3)$$

Now, $\sum_{j=0}^{k} \frac{1}{n-j} < \int_{n-k-1}^{n} \frac{1}{x} dx = \ln(\frac{n}{n-k-1})$. Therefore,

$$\mu(k,n,\lambda) < \frac{1}{\lambda} \ln(\frac{n}{n-k-1}) \qquad (4)$$

By examining (1)-(4) in turn, we can now consider how the reliability of the systems in our examples is affected by an increase in their fault-tolerance (and size).

Consider a $(k,k+1)$-resilient system $(k > 0)$. By (1),

$$F(t;k,k+1,\lambda) = (1 - e^{-\lambda t})^{k+1}$$

$$< (1 - e^{-\lambda t}) = F(t;0,1,\lambda) \qquad \text{for all } t ,$$

where $F(t;0,1,\lambda)$ is the probability of failure for one processor. Hence, a $(k,k+1)$-resilient system has lower probability of failure than one processor for all time. This is not true in general.

Graph 1 plots the probability of failure $F(t;k,n,1)$ for some $(k,3k+1)$-resilient systems. (Note: From (3), the MTTF for one processor is $\mu(0,1,\lambda)=\frac{1}{\lambda}$. By choosing $\lambda=1$, we in effect choose as the unit of time the MTTF of one processor.) The graph shows that, for $k > 0$, although the probability of failure for $(k,3k+1)$-resilient systems is initially less than that of one processor, it is greater eventually. We say that such systems have a higher probability of failure *in the long run*. Moreover, the larger the system, the faster the increase in probability from 0 to 1 (see Graph 1 for k=10, n=31).

This effect of size can be clearly seen from Graph 2, which plots the density function $f(t;k,n,1)$ for the same systems. It shows that the peak in the density curve is sharper for larger systems, so that the period in which the system will fail is very well-defined for large systems. Furthermore, the density curve for a (10,31)-resilient system drops to 0 before t=1.0, which is the MTTF of one processor. This implies that the MTTF of a (10,31)-resilient system is less than that of one processor.

Graph 3 indicates the combination of k and n values that makes the MTTF of a (k,n)-resilient system greater than that of one processor. Also plotted are the relationships between k and n in our examples. Note that none of the aforementioned combinations satisfy these relationships, i.e. in all four examples, a multi-processor system has shorter MTTF than one processor. Furthermore, suppose n is more than linear in k, say $n=g(k)$, where $\lim_{k\to\infty}\frac{k}{g(k)}=0$ (e.g. $g(k)=2^{k+1}$). Then, $\ln(\frac{n}{n-k-1}) = \ln(1/(1-\frac{k+1}{g(k)}))\to 0$ as $k\to\infty$, so (4) implies that the MTTF of such systems tends to 0 as k increases.

For what values of n would a (k,n)-resilient system have a MTTF larger than a processor's MTTF? By (4), if $\mu(k,n,\lambda) > \mu(0,1,\lambda)=\frac{1}{\lambda}$, then $\ln(\frac{n}{n-k-1}) > 1$, or $n < \frac{e}{e-1}(k+1) < 1.6(k+1)$. Hence, n must satisfy

$$k < n < 1.6k + 1.6 \qquad (5)$$

Now, if $F(t;k,n,\lambda) < F(t;0,1,\lambda)$ for all t, then $\mu(k,n,\lambda) > \mu(0,1,\lambda)$. Therefore, (5) is also a necessary condition for a (k,n)-resilient system to have a smaller probability of failure (than one processor) in the long run.

We now discuss the significance of these remarks in the context of replicated systems. A common way of implementing fault-tolerant computing is to have several processors perform the same computation, then vote on the result. For SIFT [14], in particular, the input to the computation is distributed by Byzantine agreement (to guard against failures during the distribution). This strategy satisfies SIFT's requirements because the designers were only interested in reducing the probability of failure in the short run. In the long run, however, such a strategy is futile because it increases the probability of failure, and reduces the mean time-to-failure. (The system is at most $(k,2k+1)$-resilient because of the voting.)

3 Conclusions

A (k,n)-resilient system that does not satisfy (5) has a smaller mean time-to-failure, and a higher probability of failure in the long run, than one processor. That replicated systems have these properties is in fact well known for voting hardware [13], and one should keep it in mind when proposing software for resilient systems. One should also bear in mind that (k,n)-resilient systems fail rapidly if n is more than linear in k. The highly restricted range of n in (5) indicates that designing (k,n)-resilient systems that would reduce the probability of failure in the long run will be difficult. However, we have assumed that faulty processors are not repaired or replaced. This is realistic for applications such as spaceborne systems, but not in general. For long run reliability, faults must (where possible) be detected and removed, and for this, the fact that large systems fail in a well-defined period may help in the design of an error detector.

Acknowledgement I thank Nat Goodman, Vassos Hadzilacos, Nancy Lynch and Ray Strong for helpful discussions.

References

[1] Bracha, G. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. Technical Report TR 84-590, Dept. of Computer Science, Cornell University (Jan. 1984).

[2] Dolev, D. The Byzantine Generals strike again. *Journal of Algorithms 3*, 1 (1982), 14-30.

[3] Dolev, D., Halpern, J., Simons, B., and Strong, H.R. A new look at fault tolerant network routing. *Proc. 16th Annual ACM Symposium on Theory of Computing*. Washington, D.C. (Apr. 1984), 526-535.

[4] Dolev, D., Lynch, N., Pinter, S.S., Stark, E.W., and Weihl, W.E. Reaching approximate agreement in the presence of faults. *Proc. 3rd IEEE Symposium on Reliability in Distributed Software and Database Systems* (1983).

[5] Dolev, D., Meseguer, J., and Pease, M.C. Finding safe paths in a faulty environment. *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Ottawa, Canada (1982), 95-103.

[6] Drenick, R.F. The failure law of complex equipment. *J. Soc. Indust. Appl. Math. 8*, 4 (Dec 1960), 680-692.

[7] Lamport, L. The Weak Byzantine Generals problem. *J. ACM 30*, 3 (July 1983), 668-676.

[8] Lamport, L., and Melliar-Smith, P.M. Byzantine clock synchronization. SRI International Report (Feb. 1984).

[9] Lamport, L., Shostak, R., and Pease, M. The Byzantine Generals problem. *ACM Transactions on Programming Languages and Systems 4*, 3 (July 1982), 382-401.

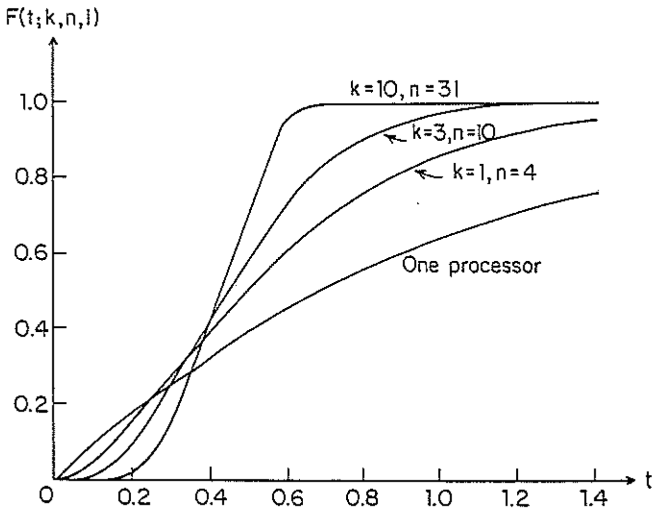
[10] Mohan, C., Strong, H.R., and Finkelstein, S. Method for distributed transaction commit and recovery using Byzantine agreement within clusters of processors. IBM Research Report RJ 3882, San Jose (June 1983).

[11] Pease, M., Shostak, R., and Lamport, L. Reaching agreement in the presence of faults. *J. ACM 27*, 2 (Apr. 1980), 228-234.

[12] Rau, J.G. *Optimization and probability in systems engineering.* Van Nostrand Reinhold, New York (1970).

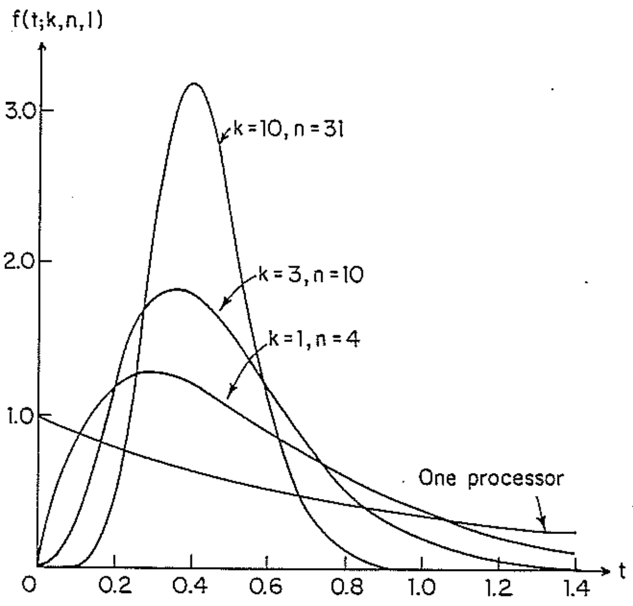
[13] Siewiorek, D.P., and Swarz, R.S. *The theory and practice of reliable system design*. Digital Press (1982).

[14] Wensley, J.H., Lamport, L., Goldberg, J., Green, M., Levitt, K.N., Melliar-Smith, P.M., Shostak, R.E., and Weinstock, C.B. SIFT : Design and analysis of a fault-tolerant computer for aircraft control. *Proc. of the IEEE 66*, 10 (Oct. 1978), 1240-1255



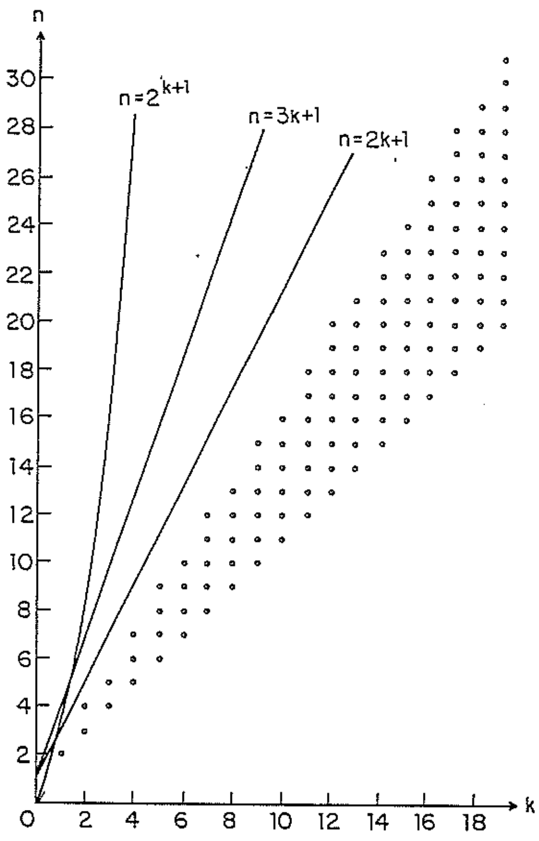
Graph 1

Probability of failure



Graph 2

Probability density function of time-to-failure



Graph 3

∘ indicates the combination of k and n values for which the MTTF of a (k,n)-resilient system is larger than the MTTF of one processor.