Metrics for the Evaluation of Proactive and Reactive Survivability[*]

Michael G. Merideth and Priya Narasimhan
School of Computer Science
Carnegie Mellon University
{*mgm,priya*}*@cs.cmu.edu*

1. Introduction

Current Byzantine-fault-tolerant survivable systems [5, 6] rely on strong theoretical properties to guarantee survivability. Evaluations of such systems generally focus on the performance overhead of the mechanisms in the fault-free case: a metric that, in itself, is not a good evaluator of survivability. This dearth of metrics makes the objective comparison of the survivability of different implementations of systems—even those that employ similar algorithms—nearly impossible. To solve this problem, we develop metrics to characterize and evaluate survivability. We intend to employ these metrics to evaluate survivable systems, such as the Starfish system [3], which we are currently developing.

If malicious faults propagate faster than traditional reactive mechanisms (which typically wait to detect a fault before reacting to recover from it) are able to recover the system, then, the survivability of the system may ultimately be compromised. This could lead to an $epidemic$ [4]—a situation in which the system is, as a consequence of faults, no longer able to recover. To curtail propagating faults, a system can instead employ proactive survivability [4] mechanisms. $Proactively$ survivable systems differ from $reactively$ survivable systems in that proactive systems may act (i) to increase resistance, (ii) to initiate recovery, or (iii) to adapt: before or concurrently with the recognition of a problem in the system.

There are two important categories of operation for any survivable system: (i) the fault-free case; and (ii) the faulty case, under which the system's resistance—though not necessarily its survivability—has been overcome, *i.e.*, a fault, either latent or active, now exists in the system. For the purpose of evaluation, it is useful to categorize the faulty case further into (a) proactive and (b) reactive, based on the survivability strategies employed by the system.

[*]This work is supported by the Army Research Office through grant number DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to the Center for Computer and Communications Security at Carnegie Mellon University.

2. Fault-Free Case

Under fault-free conditions, the primary aspect that is of interest in a system is its performance. The following metrics quantify the performance overhead of the survivability mechanisms in terms of measurable quantities:

- *System latency*, $e.g.$, additional message delays
- *Throughput*, $e.g.$, reduction in the number of messages per second that the system can process
- *Traffic*, $e.g.$, additional number of messages exchanged
- *Complexity*, $e.g.$, additional lines of code required
- *Redundancy*, $e.g.$, additional resources required for replication

3. Reactive Faulty Case

The metrics of Section 2 are applicable here, but would need to account for the system's performance under a fault, and as it recovers and adapts. The following additional metrics aim to quantify the survivability (rather than the performance) of a system:

- *Window of vulnerability*: The time period, introduced in [1], during which an additional fault(s) will result in an epidemic. In [1], the window of vulnerability occurs frequently—even if there are no faults—because parts of the system are periodically rebooted. More generally, a window of vulnerability is likely to occur after a fault, but before recovery is complete. The length of the window of vulnerability can be calculated based on the anticipated time involved in recovering the system. It could be measured in terms of the actual time to recovery.
- *Fault-detection latency:* The length of time to fault-recognition, from the time at which the system is compromised. This is likely to vary, based on the type and success of an attack or fault. Calculations of this metric will be dependent on the fault model, while measurements of this metric will be dependent on the

fault/attack injection mechanism. In systems that reactively recover, this metric is particularly important, as the system's recovery mechanisms are not triggered until the system recognizes that it has been compromised.

- *Recovery latency:* The time from fault-detection to the time that the fault—and its negative effects—have been removed from the system.
- *Reactive fault-detection accuracy:* The likelihood that fault detection involves neither false positives ($i.e.$, detection of faults that do not exist) nor false negatives ($i.e.$, failure to detect faults that do exist). This is adversely impacted whenever a small time-out value leads to suspicion of a fault, regardless of the fact that there is no fault in the system; in such cases, the system's performance and availability suffer.

4. Faulty Proactive Case

Proactive survivability exploits available information—about a fault and its propagation—that might be useful in preventing the fault from corrupting other parts of the system. The aim of a proactively survivable system is to ensure that its likelihood of an epidemic is lower than that of a reactively survivable system. *Proactive notification* is the receipt of a message about either (i) the detection of a fault in another part of the system or (ii) the suspicion that a fault has propagated to the recipient of the notification. As in the faulty proactive case, the metrics of Section 2 are applicable here; however, they would need to account for the performance overhead of the proactive mechanisms. The following metrics instead serve to quantify the effectiveness of the proactive mechanisms:

- *Proactive-notification latency:* The delay between the detection of a fault in one part of the system and the receipt of the corresponding proactive-notification message in another part of the system.
- *Proactive-notification accuracy:* The likelihood that proactive-notification messages involve neither false positives nor false negatives. Because proactive-notification messages may concern either an identified fault or its propagation, the proactive-notification accuracy is affected both by (i) the reactive-fault detection accuracy and (ii) the accuracy of the mechanisms that detect fault-propagation.
- *Proactive bonus:* The benefit of proactively containing the propagation of faults, instead of waiting until they are reactively detected. This benefit can be described in terms of a reduced window-of-vulnerability, or in terms of the additional time to increase resistance or to adapt to the fault. The proactive bonus can be calculated as the period between (i) the time of proactive-notification concerning an impending/undetected fault; and (ii) the time of fault-detection, if that part of the system were to rely only on reactive fault-detection. With a proactive $containment$ strategy, in the ideal case, notification will occur before the system sustains additional faults. With a proactive $recovery$ strategy, recovery will be initiated before the fault has sufficient time to manifest itself through errors.

5. Related Work

In the BFT system [1], the window of vulnerability is affected by both a tunable time-out length and the recovery time of each node. The evaluation of systems based on BFT focuses on the performance overhead in the case where recovery time is fixed. ITDOS [6] and Immune [5] rely on the properties of their underlying secure group communication systems, in order to provide survivable infrastructural (middleware) support. Neither provides an evaluation of the observed survivability of the system in the faulty cases.

SEI's Survivable System Analysis method [2] represents a different and complementary methodology for evaluating system survivability. SSA evaluates survivability properties based on the intended environment and the anticipated capabilities of likely attackers. While SSA is focused on analysis at the architectural level, our metrics are intended for the run-time evaluation of implementations of survivable systems.

References

[1] M. Castro and B. Liskov. Proactive recovery in a Byzantine-fault-tolerant system. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI '00)*, 2000.

[2] R. J. Ellison, R. C. Linger, T. Longstaff, and N. R. Mead. Survivable network system analysis: A case study. *IEEE Software*, 16(4):70–77, July/August 1999.

[3] K. P. Kihlstrom and P. Narasimhan. The Starfish system: Providing intrusion detection and intrusion tolerance for middleware systems. In *IEEE Workshop on Object-oriented Real-time Dependable Systems*, 2003.

[4] M. G. Merideth and P. Narasimhan. Proactive containment of malice in survivable distributed systems. In *Proceedings of the 2003 International Conference on Security and Management (SAM'03)*, 2003.

[5] P. Narasimhan, K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. Providing support for survivable CORBA applications with the Immune system. In *International Conference on Distributed Computing Systems*, pages 507–516, 1999.

[6] D. Sames, B. Matt, B. Niebuhr, G. Tally, B. Whitmore, and D. Bakken. Developing a heterogeneous intrusion tolerant CORBA system. In *Proceedings of the International Conference on Dependable Systems and Networks, 2002 (DSN '02)*, pages 239–248, 2002.