1 Complete Experimental Evaluation

Case Study	Redundancy	Masking Distance	Time	Time(Det)
Redundant Memory Cell	3 bits	0.333	$0.7s$	$0.6s$
	5 bits	0.25	$2.5s$	$1.9s$
	7 bits	0.2	$7.2s$	$5.7s$
	9 bits	0.167	$1m.4s$	$1m11s$
	11 bits	0.143	$28m27s$	$26m10s$
N-Modular Redundancy	3 modules	0.333	$0.6s$	$0.5s$
	5 modules	0.25	$1.2s$	$0.7s$
	7 modules	0.2	$5.6s$	$3.8s$
	9 modules	0.167	$2m55s$	$2m32s$
	11 modules	0.143	$75m17s$	$72m48s$
Dining Philosophers	2 phils	0.5	$0.6s$	$0.6s$
	3 phils	0.333	$1.9s$	$0.9s$
	4 phils	0.25	$5.9s$	$2.6s$
	5 phils	0.2	$25.3s$	$24.1s$
	6 phils	0.167	$19m.23s$	$11m39s$
Byzantine Generals	3 generals	0.5	$0.9s$	–
	4 generals	0.333	$17.1s$	–
	5 generals	0.333	$429m54s$	–
Raft LRCC (5)	1 follower	0	$0.7s$	$0.8s$
	2 followers	0	$5.6s$	$3.6s$
	3 followers	0	$49m.50s$	$37m.53s$
BRP(1)	1 retransm.	0.333	$0.7s$	–
	5 retransm.	0.143	$0.8s$	–
	10 retransm.	0.083	$1.3s$	–
	20 retransm.	0.045	$3.9s$	–
	40 retransm.	0.024	$4.8s$	–
BRP(5)	1 retransm.	0.333	$4.2s$	–
	5 retransm.	0.143	$4.8s$	–
	10 retransm.	0.083	$6.1s$	–
	20 retransm.	0.045	$8.7s$	–
	40 retransm.	0.024	$18.6s$	–
BRP(10)	1 retransm.	0.333	$4.7s$	–
	5 retransm.	0.143	$6.4s$	–
	10 retransm.	0.083	$10.1s$	–
	20 retransm.	0.045	$20.5s$	–
	40 retransm.	0.024	$1m.9s$	–

Table 1. Results of the masking distance for the case studies.

Some words are useful to interpret the results. For the case of a 3 bit memory the masking distance is 0.333, the main reason for this is that the faulty model (in the worst case) is only able to mask 2 faults (in this example, a fault is an unexpected change of a bit) before failing to replicate the nominal behaviour (i.e.

reading the majority value). Thus, the result comes from the definition of masking distance and taking into account the occurrence of two faults. The situation is similar for the other instances of this problem with more redundancy.

N-Modular-Redundancy consists of N systems, in which these perform a process and that results are processed by a majority-voting system to produce a single output. Assuming a single perfect voter, we have evaluated this case study for different numbers of modules. Note that the distance measures for this case study are similar to the memory example.

For the dining philosophers problem we have adopted the odd/even philosophers implementation (it prevents from deadlock), i.e., there are $n-1$ *even* philosophers that pick the right fork first, and 1 *odd* philosopher that picks the left fork first. The fault we consider in this case occurs when an even philosopher behaves as an odd one, this could be the case of a byzantine fault. For two philosophers the masking distance is 0.5 since a single fault leads to a deadlock, when more philosophers are added this distance becomes smaller.

Another interesting example of a fault-tolerant system is the Byzantine generals problem, introduced originally by Lamport et al. [3]. This is a consensus problem, where we have a general with $n-1$ lieutenants. The communication between the general and his lieutenants is performed through messengers. The general may decide to attack an enemy city or to retreat; then, he sends the order to his lieutenants. Some of the lieutenants might be traitors. We assume that the messages are delivered correctly and all the lieutenants can communicate directly with each other. In this scenario they can recognize who is sending a message. Faults can convert loyal lieutenants into traitors (byzantines faults). As a consequence, traitors might deliver false messages or perhaps they avoid sending a message that they received. The loyal lieutenants must agree on attacking or retreating after $m+1$ rounds of communication, where m is the maximum numbers of traitors. Here we consider the case of $m=1$.

Raft [4] is a consensus algorithm that has become somewhat popular in the last years. In Raft, a leader is elected from the available servers, once elected, the leader receives requests from clients and sends them to its followers so they can replicate the new information on their states. A follower may reject an entry if there is an inconsistency between its log and the leader's log, this is considered a fault. When this occurs, the leader tries again with the previous entry on its log, this is repeated until a match occurs. Here we model only one of the different stages of Raft algorithm, namely Log Replication. Specifically, we adapt the consistency check made by the protocol to ensure the consistency between the followers' logs and the log of the leader. We computed the masking distance for this model with 1, 2 and 3 followers, and considering a leader's log of 5 entries. Let us note that this distance is always 0. Intuitively, this happens because, even though there could be some rejections, eventually the logs will agree (in the worst case, the protocol will discard all the entries of the followers' log reaching the empty log, and then it will copy back the entries of the leader to the other logs).

The Bounded Retransmission Protocol (BRP) is a well-known industrial case study in software verification. While all the other case studies were treated as

toy examples and analyzed with δ_m (see [1]), the BRP was modeled closer to the implementation following [2], considering the different components (sender, receiver, and models of the channels). To analyze such a complex model we have used instead the weak masking distance δ_m^W (see [1]). We have calculated the masking distance for the bounded retransmission protocol with 1, 5 and 10 chunks, denoted BRP(1), BRP(5) and BRP(10), respectively. We observe that the distance values are not affected by the number of chunks to be sent by the protocol. This is expected because the masking distance depends on the redundancy added to mask the faults, which in this case, depends on the number of retransmissions.

References

1. Pablo F. Castro, Pedro R. D'Argenio, Ramiro Demasi, and Luciano Putruele. Measuring masking fault-tolerance. In *TACAS 2019, Prague, Czech Republic*, 2019.
2. Jan Friso Groote and Jaco van de Pol. A bounded retransmission protocol for large data packets. In *Algebraic Methodology and Software Technology, 5th International Conference, AMAST '96, Munich, Germany, July 1-5, 1996, Proceedings*, pages 536–550, 1996.
3. Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
4. Diego Ongaro and John K. Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319. USENIX Association, 2014.