On the Generalization of $n > k * t$ *
(Brief Announcement)

Flavio P. Junqueira
flavio@cs.ucsd.edu

Keith Marzullo
marzullo@cs.ucsd.edu

University of California, San Diego
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA

1 Introduction

Most fault-tolerant algorithms are designed using a *threshold model*: one assumes that out of n components, no more than t can be faulty. For example, solutions to the Consensus problem are usually developed assuming no more than t of the n processes are faulty where "being faulty" is specialized by a failure model. It is a convenient model to make. For example, bounds are easily expressed as a function of t: if processes can fail in an arbitrary manner, then without using digital signatures the Consensus problem requires $n > 3 * t$ processes [1]. But, such bounds are most useful when processes have identical probabilities of failure and they fail independently.

If this is not the case (for most systems of interest today, it rarely is), then t needs to be large enough to cover the number of failures in any run of the system. If one instead takes into account how processes fail, then one can have more efficient protocols, both in terms of running time and of replication. Thus, we have been studying various distributed problems in the context of dependent process failures [2].

So far, we have been concentrating on the Consensus problem, and we have developed new bounds and protocols from first principles. This exercise has led to the question of whether one can instead develop bounds and protocols using the results that have been previously developed for the threshold model. This brief paper describes our progress towards answering this question. We have generalized the assumption $n > k * t$ for integer $k \geq 1$ to our dependent failure model and have used it to generalize lower bound results. We have a conjecture for how to generalize upper bound results.

*This work was developed in the context of RAMP, which is DARPA project number N66001-01-1-8933.

2 System Model

We assume a message-passing distributed system: there is a set Π of processes interconnected by message channels. In such systems, processes communicate only by exchanging messages. Because we focus on process failures, we ignore channel failures and so simply consider them to be reliable.

In our system model, we allow processes to fail dependently. To model failure correlations, we use *cores* and *survivor sets*, which are abstractions that enable one to represent concisely such correlations [2]. Informally, a core is a reliable subset of processes: in every execution of the system, there is one correct process in every core. Survivor sets are subsets of processes that have at least one element from every core. Thus, in every execution of the system, there is at least one survivor set containing only correct processes.

More formally, let R be a rational number expressing a desired reliability, and $r(x)$, $x \subseteq \Pi$, be a function that evaluates to the reliability of the subset x. We define cores and survivor sets as follows:

Definition 2.1 Given a set of processes Π and rational target degree of reliability $R \in [0, 1]$, the set of processes c is a *core* of Π if and only if: (a) $c \subseteq \Pi$; (b) $r(c) \geq R$; (c)$\forall p \in c$, $r(c - \{p\}) < R$. Given a set of processes Π and a set of cores C_Π, s is a survivor set if and only if: (a) $s \subseteq \Pi$; (b) $\forall c \in C_\Pi$, $s \cap c \neq \emptyset$; (c) $\forall p_i \in s$, $\exists c \in C_\Pi$ such that $p_i \in c$ and $(s - \{p_i\}) \cap c = \emptyset$. C_Π and S_Π denote the set of cores and of survivor sets of Π, respectively.

One can also define cores and survivor sets without resorting to R. For example, one can use attributes to correlate failures [3]. Systems that share attributes, such as the same software packages, are likely to share vulnerabilities that can lead to them failing together. In the remaining of this paper, we assume that cores and survivor sets are computed in some manner and described using a *system configuration* $\langle \Pi, C_\Pi, S_\Pi \rangle$.

3 Generalizing $n > k * t$

Many lower bounds on process replication for problems in distributed computing are expressed as an inequality $n > k*t$, where n is the number of processes, k is an integer replication factor, and t is the maximum number of process failures to be tolerated. A common strategy used to arrive at such a bound is to partition the processes into k equally-sized subsets. Then, one constructs k executions, where in each subset all of the processes in one of the subsets fail. There will be one execution that violates some property of interest. For example, the Consensus problem with arbitrary failures and with no digital signatures has the well-known lower bound on process replication $n > 3 * t$ [1]. The proof consists of partitioning $3 * t$ processes into subsets $\{A, B, C\}$, each one containing t processes, and showing that two valid executions implies a third that violates agreement. Thus, any generalization of $n > k * t$ needs to consider partitionings of the processes into subsets that don't contain cores.

Many upper bounds are also expressed using the same inequality. Such bounds are often derived by considering sets of processes that can interact. For example, algorithms for problems in distributed computing often have a process take an action only after it has received a message from a certain number of processes. For example, the $\diamond\mathcal{S}$ Consensus algorithm by Chandra and Toueg requires $n > 2 * t$ [4]. In this algorithm, a coordinator broadcasts a *decide* message once a majority of the processes adopt its estimate. Thus, a generalization of the $n > k*t$ needs to consider survivor sets, since these are minimal subsets of processes that are correct in at least one execution.

We now state two equivalent properties that both generalize the $n > k*t$ replication requirement for integer $k \geq 1$. These properties are based on our two previous observations. Consider the following properties for a system $\langle \Pi, C_\Pi, S_\Pi \rangle$:

Property 3.1 k-Partition
For every partition $A = \{A_1, A_2, \cdots, A_k\}$ of Π, there is a subset $A_i \in A$ such that A_i contains a core. $\Box$

Property 3.2 k-Intersection
For every subset $\{s_1, s_2, \cdots, s_k\} \subseteq S_\Pi$, $(\cap_i s_i) \neq \emptyset$. $\Box$

Theorem 3.3 *k-Partition $\equiv$ k-Intersection.* $\Box$

A direct implication of the k-Intersectionproperty is that the intersection of any set of $k-1$ distinct survivor sets contains a core, for $k > 1$. To see this, suppose that k-Intersectionholds. Then, for every subset $S' = \{s_1, s_2, \cdots, s_{k-1}\} \subset S_\Pi$ and survivor set $s \in S_\Pi/S'$, we have that $(\cap_i s_i) \cap s \neq \emptyset$. Since this is true for any $s \in S_\Pi/S'$, each survivor set in S' contains at least one element from each other survivor set. By assumptiom, in every execution, at least one survivor set contains only correct processes, and hence the intersection must contain a core. This implied property is a generalization of the **Byzantine Intersection** property we showed in [2]. We used this intersection property to modify an existing algorithm for the Consensus problem in synchronous systems with arbitrary failures.

We now argue that k-Partitionis a lower bound on process replication for any problem that requires $n > k*t$ process replication, for an integer $k \geq 1$. Let P be some problem in fault-tolerant distributed computing that requires $n > k*t$ replication for some integer $k \geq 1$. This threshold implies that a system requires at least $(k*t)+1$ processes to solve P, assuming at most t failures. From the pigeonhole principle, any partition of the processes into k subsets requires that one subset contains at least $t+1$ processes. This subset therefore contains a core and the system consequently satisfies k-Partition.

Consider now an algorithm under the threshold model that requires $n > k*t$ replication. In addition, suppose there is at least one statement in this algorithm in which a process sends or receives $n-(\gamma*t)$ messages m, for some positive integer $\gamma < k$. We conjecture that we can automatically translate this algorithm to our model by simply replacing every statement in which some process sends or receives $n-(\gamma*t)$ messages by one in which a process sends or receives messages to elements in some intersection of γ survivor sets. Although we have not been able to prove a general result, this conjecture holds for the examples we have looked at. We discuss one of these examples in the following section.

4 An Example: State-machine Replication

State-machine replication is a well-known approach for implementing fault-tolerant distributed services [5]. Typically, this approach is characterized by clients that send requests to a set of server replicas executing a deterministic state machine. A correct server processes an agreed-upon sequence of requests and returns the corresponding sequence of responses to the clients. Because every server executes the same deterministic state-machine and the replicas agree on the sequence of commands, the replies from all the correct servers are the same. Thus, after collecting answers from a subset of these servers, a client computes an output.

Assuming arbitrary failures, the minimum number of servers needed to implement state-machine replication is $2t + 1$ in the threshold model [5]. The outcome decided by the client is given by the value given in a majority of the replies.

In our model of dependent failures, one can implement a service based on state-machine replication by using the same approach, but assuming 2-Intersection instead of $n > 2 * t$ as the process replication requirement. The client waits until it receives identical replies from a survivor set instead of from a majority $(n - t)$ as in the threshold model. From the previous section, if a system satisfies 2-Intersection, then the intersection between every pair of survivor sets is not empty. As a consequence, we have that every survivor set has to contain a correct process, because we have by assumption that in every execution at least one survivor set contains only correct processes. Thus, it is impossible for a client to receive one reply from one survivor set and a different reply from another survivor set.

5 Conclusions and Future Work

Cores and survivor sets constitute a concise way of representing correlation of failures among processes. There are several benefits of using this model, such as a reduced number of rounds and a reduced number of replicas. In this paper, we have discussed another benefit of considering such model: being able to take advantage of the benefits of modeling with cores and survivor sets without designing new algorithms from scratch. This is a first step towards a generalization of the threshold model for dependent process failures.

We have proofs for most of the assertions made throughout the paper. We have yet to show the conjecture presented in Section 3, which refers to the automatic translation of algorithms for a particular class of algorithms. In order to achieve this goal, we are looking into new problems to gain intuition. If we can show this (or perhaps a more restricted) conjecture, then we believe it could lead to a widespread practical application of cores and survivor sets.

References

[1] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[2] F. Junqueira and K. Marzullo, "Synchronous Consensus for Dependent Process Failures," in *International Conference on Distributed Computing and Systems (ICDCS)*, May 2003.

[3] F. Junqueira, R. Bhagwan, K. Marzullo, S. Savage, and G. M. Voelker, "The Phoenix Recovery System: Recovering from the ashes of an Internet catastrophe," in *IX Hot Topics in Operating Systems (HotOS-IX)*, May 2003.

[4] T. Chandra and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems," *Journal of the ACM*, vol. 43, pp. 225–267, March 1996.

[5] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: a tutorial." ACM Computing Surveys, December 1990.