The Parsimonious Approach to Constructing Fault-Tolerant Protocols

HariGovind V. Ramasamy[†], Christian Cachin[†], Adnan Agbaria[‡], and William H. Sanders[††]

[†]IBM Zurich Research Laboratory, Rüschlikon, Switzerland
{hvr,cca}@zurich.ibm.com

[‡]USC Information Sciences Institute - East, Arlington, VA 22203, USA
adnan@isi.edu

[††]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
whs@uiuc.edu

1 Introduction

Fault-tolerant distributed protocols, by definition, are designed with the worst in mind. This focus on tolerating the worst often leads to expensive designs that overlook the practical observation that the occurrence of disruptions and failures is rare relative to the lifetime or mission periods of many systems. The class of optimistic fault-tolerant protocols leverages that observation and strives to achieve efficiency during normal operation of the system. In this paper, we describe a specialization of the optimistic approach to building fault-tolerant protocols that we call the *parsimonious* approach.

In the parsimonious approach, we design the protocol with the explicit aim of achieving frugality or efficiency with respect to a given metric of interest $\mathcal{M}$ (such as latency degree, resource usage, message complexity, etc.) while never violating correctness (i.e., *safety* and *liveness*). When certain operational assumptions are satisfied, the design uses some lightweight mechanism that can provide desired protocol functionality with optimal $\mathcal{M}$. The optimistic hope is that those assumptions are satisfied more often than not, i.e., the chosen parsimonious mechanism is applicable for most of the system's lifetime. For this hope to be realistic, however, the operational assumptions of the mechanism must have good coverage. The protocol design uses a more expensive fall-back or recovery mechanism whenever the assumptions are not satisfied; after ensuring correctness properties, the protocol then reverts back to using the parsimonious mechanism. To handle situations that are contradictory to the assumptions implies the detection of their occurrence in the first place, i.e., failure or anomaly detection [1]. Correctness must never be violated despite imperfections in the detection mechanism.

Parsimonious Mode

Optimistically hope that conditions are "normal."
For a given metric of interest, $\mathcal{M}$, use a mechanism that
- when the hope is met, provides protocol functionality with *optimal* $\mathcal{M}$, and
- when the hope is not met, *may not make progress* but *never violates safety*.

Eventually detect inability to make progress; switch to recovery mode.

Recovery Mode

Use a fall-back mechanism to ensure progress.
Factor in latest failure detection information; redefine "normal" conditions.
Switch back to parsimonious mode for next epoch.

Figure 1. Protocol Structure

2 Structure of Parsimonious Protocols

A parsimonious protocol operates in epochs, with each epoch consisting of a parsimonious mode and a recovery mode (Figure 1). The concept of epochs takes into account the behavior of many real systems that alternate between long periods of stability and relatively short periods of instability. The strategy is to roughly align the stable periods with the parsimonious modes of protocol operation, so that the protocol operates in the parsimonious mode most of the time.

For a given metric of interest $\mathcal{M}$, the parsimonious mode employs a lightweight or parsimonious mechanism that, under normal conditions, provides the protocol functionality with optimal $\mathcal{M}$. The specification of "normal" conditions (as opposed to faulty or unstable conditions) embodies the operational assumptions under which the mechanism is applicable. That specification will vary depending on the mechanism and the potential deployment setting (e.g., LAN or Internet). For a certain parsimonious mechanism, "normal conditions" may refer to the state in which the underlying network is relatively stable in terms of the message transmission delays. For another mechanism, "normal conditions" may refer to the status of the system in which no additional faults other than the ones already known have occurred.

Under abnormal conditions that are indicative of faults or instability, the technique employed in the parsimonious mode may not be able to make progress, but should never violate safety. To ensure progress, it is important to detect the presence of abnormal conditions and then recover. For eventual progress, detection mech-

Protocol	Metric of Interest ($\mathcal{M}$)	Optimal $\mathcal{M}$	Parsimonious Mechanism	Recovery Mechanism
async. atomic broadcast	latency degree	2	leader-initiated *short* consistent broadcast [2] for each payload broadcast	randomized validated Byzantine agreement
async. atomic broadcast	message complexity	$\mathcal{O}(n)$	leader-initiated *strong* consistent broadcast [2] for each payload broadcast	randomized validated Byzantine agreement
async. request execution	overall resource usage	$t+1$	only a subset of $t+1$ replicas execute requests	*all-active execution*: all replicas execute request

Table 1. Example Parsimonious Protocols [2]

anisms with relatively weak properties (e.g., the class of *eventually strong* [1] failure detectors) will suffice.

When detection mechanisms signal the presence of abnormality, the protocol switches to a more expensive recovery mode, which ensures that progress is eventually made. Soon after making some progress and thereby ensuring the protocol's continued provision of correctness, the protocol switches back to the parsimonious mode for the next epoch. Before the start of the new epoch, the protocol must factor in the information revealed by the detection mechanism in the last epoch and make appropriate changes that essentially redefine what constitutes normal conditions. These changes may include, for example, the quarantine of a process, the activation of a backup, the use of an alternative message delivery path, and so forth. If the cause for the abnormal condition was a transient fault, then perhaps the condition will have disappeared by the time the protocol starts the new epoch. However, in the absence of reliable detection information as to whether the abnormal condition is transient or permanent, such changes are necessary to allow the protocol to make progress in the parsimonious mode of the new epoch. Otherwise, the protocol will make progress only when it switches to the recovery mode in each epoch, resulting in a rather inefficient protocol.

3 Experiences in Applying the Parsimonious Approach

Consider a Byzantine-fault-tolerant state-machine-replicated service consisting of n replicas and tolerant to t faulty ones. To maintain state consistency across all correct replicas of the service, there must be two phases [3]: an *agreement phase* in which the replicas perform Byzantine agreement or atomic broadcast to agree on the order of state machine operations to execute, and an *execution phase*, in which the replicas execute the requests in the agreed-upon order. In this context, we have designed three parsimonious protocols:
(1) An asynchronous protocol for the execution phase that is parsimonious in the overall amount of resources (across all replicas) used for request execution.
(2) An asynchronous atomic broadcast protocol for the agreement phase that is parsimonious in *message complexity*, i.e., the number of protocol messages that correct replicas generate per atomically delivered payload.
(3) An asynchronous atomic broadcast protocol for the agreement phase that is parsimonious in *latency degree*, i.e., the number of communication steps involved per atomically delivered payload.

Table 1 gives a summary of the protocols. The detailed description for the protocols and proofs of correctness appear in [2]. We have implemented and experimentally evaluated the above protocols individually and collectively in the context of implementing a prototype replicated service on both LAN and WAN settings [2]. The results confirmed the superior efficiency characteristics of protocol operation in the parsimonious mode and the continued provision of correctness despite faults, unstable conditions, and unreliable failure detection.

Although the parsimonious approach guarantees protocol correctness despite unreliable detection mechanisms, the overall efficiency characteristics are, in large part, influenced by how quickly and accurately the mechanisms signal the occurrence of faults or unstable conditions. Constructing fault or anomaly detection mechanisms requires an understanding of the normal system operational characteristics. That task is relatively easy for static systems with extensive recorded operational history, but may be very challenging for systems that are dynamic or whose behavior evolves over time. For the latter kind of systems, the specification of what constitutes "normal" operational characteristics may vary over time, and it is important to adapt the detection mechanisms accordingly.

References

[1] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2):225–267, 1996.

[2] H. V. Ramasamy. *Parsimonious Service Replication for Tolerating Malicious Attacks in Asynchronous Environments*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[3] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating Agreement from Execution for Byzantine Fault Tolerant Services. In *Proc. 19th ACM Symp. on Operating Systems Principles*, pages 253–267, Oct 2003.