Probabilistic Adaptive Time-Aware Consensus

Mônica Dixit António Casimiro Paulo Verissimo
{mdixit,casim,pjv}@di.fc.ul.pt
University of Lisboa*

1 Introduction

The consensus problem is a fundamental building block on the design of distributed systems, as it contributes to the coordination of actions in order to achieve consistent decisions. In a consensus execution, each process proposes an initial value to the others, and, despite failures, all correct processes have to agree on a common value, which has to be one of the proposed values [3]. The solution for many agreement problems, such as atomic broadcast, leader election or clock synchronization, relies on the ability to achieve some form of consensus among a set of processes. In distributed applications that require agreement-based services to be provided with certain guarantees with respect to timeliness, it may be necessary to ensure not only that consensus is solvable, but also that it is solvable within a bounded amount of time. In other words, it may be necessary to solve a *timed consensus* problem.

However, most of the existing literature on consensus is focused on asynchronous system models [2], in which consensus terminates eventually, as soon as allowed by the environment. Moreover, distributed systems are required to operate in settings that are every day more dynamic. There is a inherent uncertainty on these environments that makes very hard (sometimes impossible) guaranteeing that consensus will always be achieved within a given static bound. If the expected execution time must be adjusted according to the environment dynamics, then applications should at least be aware of how much time the protocol execution takes under a set of specific conditions. This feature is particularly useful for real-time systems, in which tasks' scheduling is defined *a priori*, in order to guarantee that the deadlines are met. In real time systems that need to execute consensus, it is fundamental to know the worst-case execution time (WCET) of the consensus protocol. These systems need to be aware of WCET changes, in order to verify if the current scheduling is still valid (deadlines remain guaranteed) or if a new one must be calculated.

We will address this problem under a probabilistic perspective. Our goal is to define, implement and evaluate a *probabilistic adaptive time-aware consensus service*.

*Faculdade de Ciências da Universidade de Lisboa. Bloco C6, Campo Grande, 1749-016 Lisboa, Portugal. Navigators Home Page: http://www.navigators.di.fc.ul.pt.

2 Probabilistic Adaptive Time-Aware Consensus Service

The main objective of this work is not limited to defining another consensus protocol, but proposing a innovative integrated solution to be provided as a service to different applications. Figure 1 shows a simplified view of the architecture that we consider in our proposal. There are two main parts: a bottom part correspond to middleware for monitoring the state of the execution environment, in which several techniques can be plugged in. This will be further explained in Section 3. On the upper part reside the services or applications that make use of this middleware, such as the time-aware consensus service that we present in the reminder of this section.

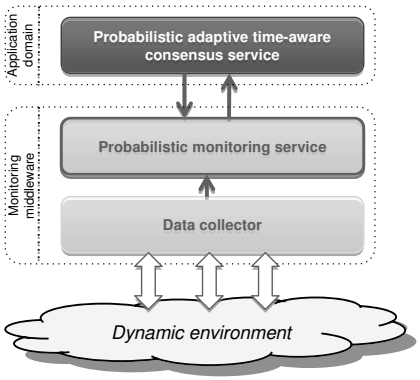


Figure 1. Probabilistic Adaptive Time-Aware Consensus Service Architecture

We define the *probabilistic adaptive time-aware consensus* according to the following properties:

- *Validity:* if a process decides v, then v was proposed by some process;
- *Agreement:* no two correct processes decide differently;
- *Probabilistic timed termination:* every correct process decides within Δ time units with probability $P_{\Delta} \geq P_{min}$, or a new Δ' holding with probability $P_{\Delta'} = P_{min}$ is returned to the application.

The validity and agreement properties are the same as in the classic consensus problem. The difference stands in the termination property, which is redefined in terms of maximum expected execution time (Δ) and its minimum probability to be secured (P_{min}). Therefore, in our approach each process participating in a consensus execution should call a function *consensus(v, Δ, P_{min})*, where v is its initial value, Δ is the assumed maximum execution time, and P_{min} is the minimum accepted probability for Δ holding.

A preliminary high-level sketch of the consensus service is presented in Algorithms 1 and 2 (notice that this is a work in progress report). It is composed by a main thread that executes a consensus protocol (Algorithm 1), and another concurrent thread which monitors the consensus deadline. This modular approach allows for testing different consensus protocols, in order to define which one is more suitable for our purposes. We also aim at proposing new protocols or adjusting existent ones to take advantage of the knowledge of message transmission delays, for example, by adapting failure detectors timeouts.

Regarding the second thread (Algorithm 2), it guarantees the *probabilistic timed termination* property by continuously verifying the probability of terminating consensus within the expected deadline, and providing the application with a new bound whenever this probability is less than the minimum required. Thus, applications can adapt their execution as soon as different environment conditions are detected.

Algorithm 1 Probabilistic time-aware consensus protocol

Require: v, Δ, P_{min}
1: Start thread 1 {which validates timing requirements}
2: Execute a consensus protocol
3: **return** consensus result v'

Algorithm 2 Thread l: validating timing requirements

1: **while** Consensus execution has not finished **do**
2: **if** $P_{\Delta} < P_{min}$ **then**
3: Calculate a new deadline Δ' with $P_{\Delta'} = P_{min}$
4: Notify application with Δ'
5: **end if**
6: **end while**

3 Underlying Middleware

As part of our solution, we need an underlying service which must be able to monitor the environment, and estimate as dependable as possible its future behavior. This service is the *probabilistic monitoring service* (Figure 1), which was first introduced in [1] as a framework for dependable adaptation. Its purpose was helping system architects in building dependable adaptive systems in probabilistic environments.

This framework (Figure 2) was modeled as a service that: (i) accepts the history size (i.e., the number of collected samples of the random variable under observation) and the required coverage (probability that the assumed bounds hold) as dependability related parameters; (ii) reads samples (e.g. measured message transmission delays) as input; and (iii) provides, as output, a bound that should be used in order to achieve the specified coverage.

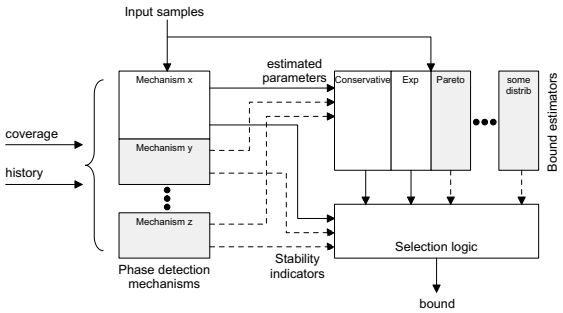


Figure 2. Schematic view of the monitoring/adaptation service.

In this previous work, we stated some assumptions about the probabilistic nature and dynamics of the monitored data, which showed to be reasonable for a large range of environments and applications. Specifically, we assume that during its lifetime a system alternates periods where its probabilistic behavior is well characterized, with transition periods where a variation of the environment conditions occurs. The proposed framework is based on the use of probabilistic methods for the recognition of the "state" of the environment and its characterization (phase detection mechanisms). Once the environment is correctly characterized (by a probabilistic distribution and its parameters), we can calculate a safe bound with the required coverage (bound estimators), in order to guide the adaptation of client applications.

References

[1] A. Casimiro, P. Lollini, M. Dixit, A. Bondavalli, and P. Verissimo. A framework for dependable qos adaptation in probabilistic environments. In *23rd ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems Track*, pages 2192–2196, Fortaleza, Ceara, Brazil, Mar. 2008.

[2] R. Guerraoui, M. Hurfin, A. Mostéfaoui, R. Oliveira, M. Raynal, and A. Schiper. Consensus in asynchronous distributed systems: A concise guided tour. In *Advances in Distributed Systems, Advanced Distributed Computing: From Algorithms to Systems*, pages 33–47, London, UK, 1999. Springer-Verlag.

[3] P. Verissimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

PROBABILISTIC ADAPTIVE TIME-AWARE CONSENSUS

Mônica Dixit, António Casimiro, Paulo Veríssimo
{mdixit,casim,pjv}@di.fc.ul.pt

Motivation

Some distributed applications require agreement-based services to be provided with certain timeliness guarantees (timed consensus).

In dynamic and uncertain environments, applications should at least be aware of consensus execution time under a set of specific conditions.

Our goal

To address this problem under a probabilistic perspective, by defining, implementing and evaluating a *probabilistic adaptive time-aware consensus service*.

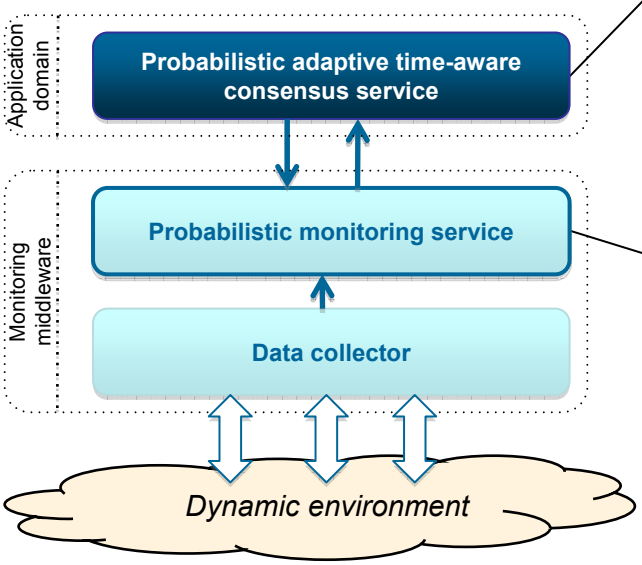
Probabilistic adaptive time-aware consensus service

Not only another consensus protocol, but a innovative integrated solution to be provided as a service to different applications.

Probabilistic adaptive time-aware consensus properties:

- **Validity:** if a process decides *v*, then *v* was proposed by some process;
- **Agreement:** no two correct processes decide differently;
- **Probabilistic timed termination:** every correct process decides within Δ time units with probability PΔ ≥ Pmin, or a new Δ' holding with probability PΔ' = Pmin is returned to the application.

Architecture



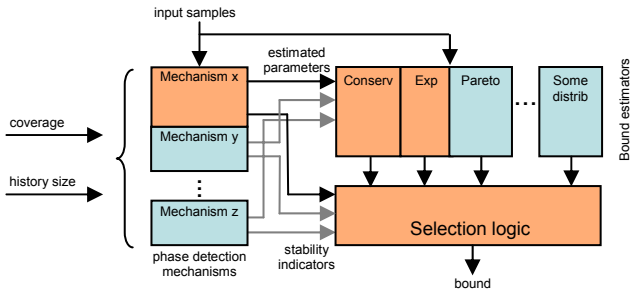
Algorithm 1 Probabilistic time-aware consensus protocol

Require: v, Δ, P_{min}

1: Start thread 1 {which validates timing requirements}
2: Execute a consensus protocol
3: **return** consensus result v'

Algorithm 2 Thread 1: validating timing requirements

1: **while** Consensus execution has not finished **do**
2: **if** $P_{\Delta} < P_{min}$ **then**
3: Calculate a new deadline Δ' with $P_{\Delta'} = P_{min}$
4: Notify application with Δ'
5: **end if**
6: **end while**



How it works

The monitoring service uses probabilistic methods to characterize the environment state and calculate a time bound.

The probabilistic adaptive time-aware consensus service uses this time bound to estimate the consensus execution time. If some consensus execution cannot terminate within the expected deadline (which may happen with a known probability), the new deadline is provided to the application.

Monitoring middleware

Probabilistic monitoring service is an underlying service to monitor the environment, and estimate as dependable as possible its future behavior.

It accepts the number of recent measured time bounds to be considered in the analysis (history size), and the minimum probability that the produced time bounds should hold (coverage) as dependability related parameters; reads measured message transmission delays as input; and provides, as output, a bound that should be used in order to achieve the specified coverage.

Current progress

The monitoring service is already implemented and its description can be found in the paper *"A framework for dependable QoS adaptation in probabilistic environments"*, published in the ACM SAC 08 conference.

The probabilistic adaptive time-aware consensus service is under development.

FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DE LISBOA
NAVIGATORS HOME PAGE: http://www.navigators.di.fc.ul.pt

This work was partially supported by the EC, through project IST-FP6-STREP-26979 (HIDENETS), and by the FCT, through the Large-Scale Informatic Systems Laboratory (LaSIGE).