The Timely Computing Base

Paulo Veríssimo
pjv@di.fc.ul.pt
FC/UL*

António Casimiro
casim@di.fc.ul.pt
FC/UL*

1. Introduction

A large number of the emerging services have interactivity or mission-criticality requirements, which are best translated into requirements for fault-tolerance and real-time.

To implement these services a given distributed systems model must be chosen. However, we have to face the following problem: fully asynchronous models do not satisfy our needs, because they do not allow timeliness specifications; on the other hand, correct operation under fully synchronous models is very difficult to achieve (if at all possible) in large-scale infrastructures, since they have poor baseline timeliness properties. Then, *what system model to use for applications with synchrony (i.e. real-time) requirements running on environments with uncertain timeliness?* We propose a framework that describes the problem in a generic way. We call it the **Timely Computing Base (TCB)** model. A detailed description of the TCB model can be found in [4].

2. Related Work

Chandra & Toueg have given a failure detector which, should the system be 'synchronous' for a long enough period, would allow to terminate consensus[1]. Cristian & Fetzer have devised the timed-asynchronous model, where the system has just enough synchronism to make decisions such as 'fail-safe shutdown'[2]. We have devised the quasi-synchronous model where parts of the system have enough synchronism to perform 'real-time actions' with a certain probability[3].

3. The Timely Computing Base

The architecture of a system with a Timely Computing Base is suggested by Figure 1. Whilst there is a generic, *payload* system over a global network, or *payload* channel, the system admits the construction of say, a *control* part, made of local TCB modules, interconnected by some form of medium, the *control* channel. Processes p execute on the several sites, making use of the TCB whenever appropriate.

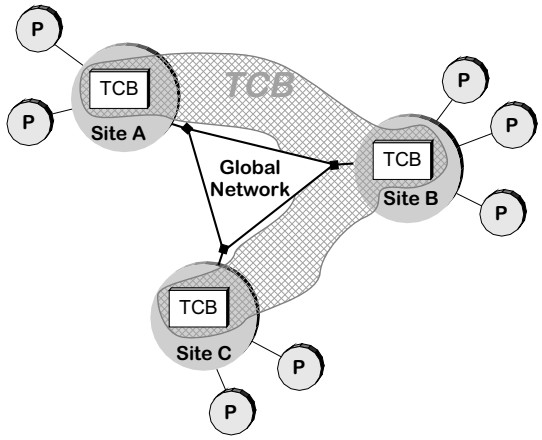

Figure 1: The TCB Architecture

3.1. Failure model

The generic payload system can have any degree of synchronism, that is, if bounds exist for processing or communication delays, their magnitude may be uncertain or not known. Local clocks may not exist or may not have a bounded rate of drift towards real time. Components *only have timing failures*— and of course, omission and crash, since they are subsets of timing failures— no value failures occur.

A relevant class of property in this context is a *timeliness property*. When timeliness properties are violated, timing failures occur.

3.2. The TCB model

The TCB subsystem, dashed in the figure, enjoys, by construction, a few synchronism properties: there exist known upper bounds on processing delays, on the rate of drift of local clocks and on message delivery delays.

These properties postulate the control of the TCB over a timely inter-TCB communication channel that may or not be based on a physically different network from the one supporting the payload channel, used for normal communication. For example, the assumption

*Faculdade de Ciências da Universidade de Lisboa (FCUL). Navigators Group Web: http://www.navigators.di.fc.ul.pt/.

of a restricted channel with predictable timing characteristics (control) coexisting with essentially asynchronous channels (payload) is feasible in some of the current networks.

A discussion about the engineering principles behind the construction of a TCB can be found in [4].

One may suggest that we are only hiding in the TCB the problem that other systems have: achieving coverage of synchrony assumptions. However, note that the TCB is a comparably small and simple part of the whole system. In consequence, enforcing strong synchronism properties for the TCB is comparably much simpler than for the whole (payload) system. Coverage of properties of the TCB can be made comparably much higher than one would achieve for the same properties in the payload system.

3.3. Services of the TCB

The TCB provides the following services: *timely execution*; *duration measurement*; *timing failure detection*. These services have a distributed scope, although they are provided to processes via the local TCB instantiations. Any service may be provided to more than one user in the system. For example, failure notification may be given to all interested users.

3.4. The several facets of timing failures

When timing failures occur, there are essentially three kinds of problems, that can arise: decreased coverage, unexpected delay and contamination.

When we make assumptions about the absence of timing failures, we have in mind a certain coverage, which is the correspondence between system timeliness assumptions and what the environment can guarantee. If the environment conditions start degrading to states worse than assumed, the coverage incrementally decreases, that is, the probability of timing failure increases. A sufficient condition for that not to happen consists in ensuring that coverage stays close to the assumed value, over an interval of mission. This notion is captured by the **Coverage Stability** property.

We define unexpected delay as the violation of a timeliness property. That can sometimes be accepted, if applications are prepared to work correctly under increased delay expectations. Finally, there can also be contamination, that we define as the incorrect behavior resulting from the violation of safety properties on account of the occurrence of timing failures. This effect has not been well understood, and haunts many designs, even those supposedly asynchronous. A sufficient condition for absence of contamination is to confine the effect of timing failures to the violation of timeliness properties alone, specified by the **No-Contamination** property.

4. Dependable and timely computing with a TCB

How can the TCB help design dependable and timely applications? A constructive approach consists in deriving sufficient conditions to solve the problems encountered because of uncertain timeliness, based on the behavior of applications and on the properties of the TCB.

Separating the mechanisms of timing failure into delay, uncoverage and contamination, allows us to introduce classes of applications that deal with combinations of the former, achieving varying degrees of dependability, when assisted by a TCB: **fail-safe**, which exhibits correct behavior or else stops in fail-safe state; **time-elastic**, which exhibits coverage stability; and **time-safe**, which exhibits no-contamination.

A generic approach to timing fault tolerance consists in mapping the application classes just described (or combinations thereof) into known fault tolerance techniques. Timing failures, in a fault-tolerance sense, can be handled in one of the following ways: masking; detection and/or recovery.

The TCB provides a framework for addressing all of these techniques, for any degree of synchronism of the payload system. We have considered[4] increasingly effective fault tolerance mechanisms, namely:

- Fail-safe operation: by switching to a fail-safe state after the first failure. Requires the timing failure detection service and applications to be of the fail-safe class;
- Reconfiguration and adaptation: by enforcing coverage stability, adapting essential timing variables to environment conditions. Requires applications to be of the time-elastic and time-safe classes.
- Timing error masking: by using replication to mask transient timing errors. Requires accurate timing failure detection and time-safety.

References

[1] Tushar Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[2] Flaviu Cristian and Christof Fetzer. The timed asynchronous system model. In *Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing*, pages 140–149, Munich, Germany, June 1998. IEEE Computer Society Press.

[3] Paulo Veríssimo and Carlos Almeida. Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models. *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS)*, 7(4):35–39, Winter 1995.

[4] Paulo Veríssimo and António Casimiro. The timely computing base. DI/FCUL TR 99–2, Department of Computer Science, University of Lisboa, April 1999. http://www.di.fc.ul.pt/biblioteca/techreports/99-2.ps.gz.