SANKHYA™

Fault Tolerance in Distributed Systems

Sankhya Technologies Private Limited.
# 30-15-58, Third Floor, "Silver Willow", Dabagardens, Visakhapatnam - 530 020 INDIA
http://www.sankhya.com

Problem Statement

Distributed embedded systems increasingly control many of today's mission critical systems. Systems ranging from wide area communication networks, command and control structures and a variety of safety critical systems depend on uninterrupted real-time behavior of their embedded subsystems for sound and reliable operation. Attaining fault-tolerance in these subsystems improves the overall availability and therefore reliability of these systems.

Solution Overview

In this document, we present a method to achieve fault tolerance in distributed embedded systems, without significantly increasing the overall complexity of the system.

Fault tolerance in a system is achieved by providing redundancy of the subsystems, the failure of which needs to be tolerated by a distributed system. For example, to provide fault tolerance in a network connection, redundant connections will have to be used.

Just providing redundancy, however does not provide fault tolerance at the application layer. In a distributed system, the pre-failure state of a faulty component will have to be preserved and operation of the system must continue from that state, using a redundant component.

Distributed systems built using OMG's Object Management Architecture, can benefit from the CORBA implementations supporting OMG's FT-CORBA specification.

Simple Fault Tolerant System using FT-CORBA Concepts

Using the full FT-CORBA specification may not be feasible for some existing distributed systems. The FT-CORBA specification is complex and implementations of this specification continue to evolve and mature.

A fault-tolerant distributed system, can be built using a few concepts of the FT CORBA specification that includes replication and state logging for recovery. The following are the FT CORBA interfaces that should be implemented to achieve fault tolerance.

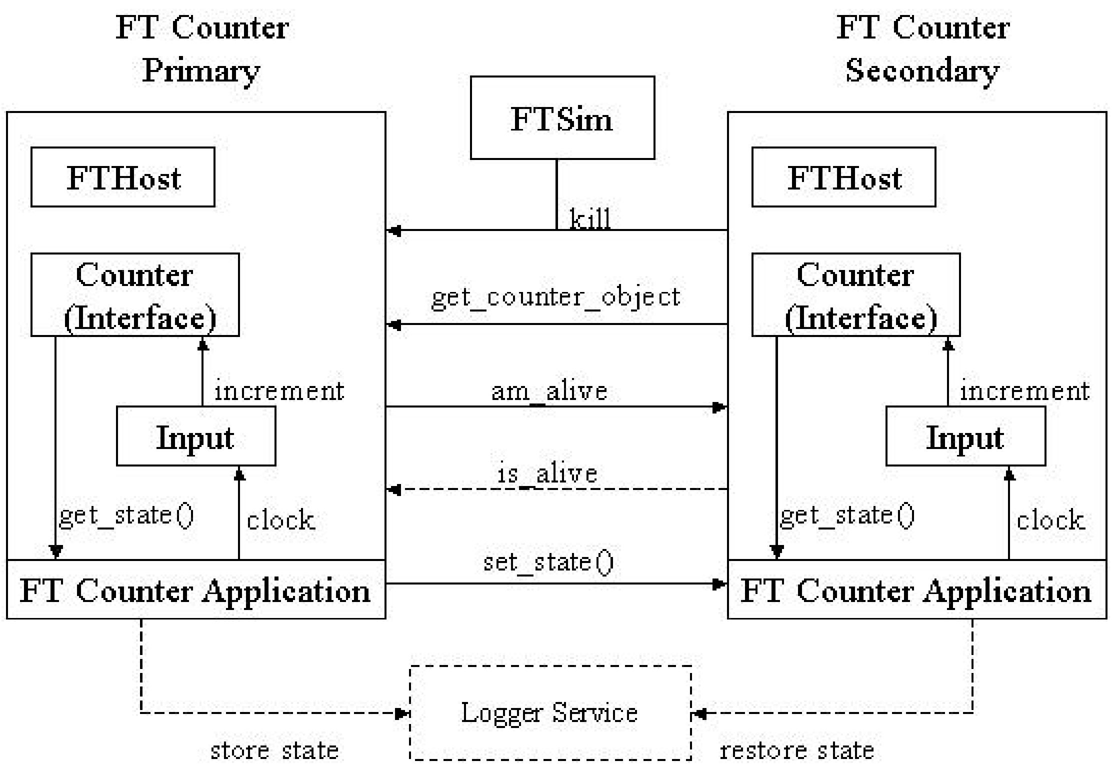
- PullMonitorable Interface
- CheckPointable Interface
- PushMonitorable Interface

Application objects that may be running on redundant subsystems can implement the above interfaces.

A simple daemon running in each redundant host can periodically poll the master copy of replicated objects using the <get_state> operation in CheckPointable interface, and synchronize all other copies of the object using <set_state>. When the primary object fails, the system can continue operating by using the replicated object. Any occurence of a fault is detected by the is_alive() operation of the PullMonitorable interface.

Such a simple implementation is not obviously ideal for all embedded systems where fault tolerance needs to be provided. However, a large majority of the systems can take advantage of this method.

Example: A Simple Fault Tolerant Counter Application



To explain fault tolerance at the application level, we consider a fault tolerant application FTCounter. We also consider two processes FTPrimary and FTsecondary each running this fault tolerant application. As shown in the diagram, the secondary is just a replica of the primary.

The counter application in the primary is started once it receives an input clock signal sent by the input interface.

```
// Input interface

interface input {
      void clock();
 };
```

The FTSecondary, a redundant but neverthless equally important component in the system, starts counting only when a fault occurs in the primary.

The counter interface implements the counter application that incrementally counts on every signal.

```
// Counter interface

interface Counter : Checkpointable {
       void SetCounter(in long init);
      void Increment();
};
```

Note that the 'Counter' interface inherits the *CheckPointable* interface. Checkpointable is a standard FT-CORBA interface that is used for state synchronization. It primarily provides *get_state* and *set_state* operations to enable the logging and recovery mechanisms to record and restore the state of the application.

The FThost interface implements the get_counter_object method that exposes the ‘Counter’ object reference for further state synchronization.

```
// FThost interface

interface fthost : processcontrol, PullMonitorable, PushMonitorable {
      Counter get_counter_object();
};
```

Note that this interface inherits two interfaces - *PullMonitorable* and *PushMonitorable*.

The PullMonitorable and PushMonitorable are standard FT-CORBA interfaces used for monitoring the health of the applications.

```
// Processcontrol interface

interface processcontrol {
     void kill();
};
```

A separate fault simulator process is used to simulate a fault in the counter application using ‘processcontrol’ interface. If the primary is terminated due to the occurrence of a fault, the secondary process which periodically monitors the primary (using PushMonitorable interface) to check for its existence, starts from the pre-fault state.

The FTprimary transfers its state to the secondary at the end of one clock input. Therefore when the secondary detects a fault in the primary, it starts counting as it already has the pre-fault state.

An alternative to using the FThost interface is to have a dedicated CORBA logging service (shown in the dotted box) that logs the state of the primary so that the secondary can get the pre-fault state from this service and start from this state. This is also called *Asynchronous State Transformation*.

Conclusion

FT-CORBA specification provides distributed embedded systems developers a standards compliant method for achieving fault tolerance. This document provides a simple method for adding fault tolerance to an application, that is standards compliant, scalable and extensible to a more elaborate fault tolerant implementation.

For Embedded and Fault-Tolerant CORBA Solutions from Sankhya, please contact **sales@sankhya.com**

SANKHYA™
Sankhya Technologies Private Limited
# 30-15-58, Third Floor, "Silver Willow",
Dabagardens, Visakhapatnam 530 020 INDIA
Tel: +91 891 5542 666 Fax: +91 891 5542 665
http://www.sankhya.com