2

Validation of Timing Properties for Interoperability in Distributed Real-Time Applications

Y. Benkhellat and J.-P. Thomesse

Centre de Recherche en Informatique de Nancy (CRIN)
2 avenue de la Forêt de Haye, 54516 Vandoeuvre-les-Nancy, France
E-mail: benkhell@loria.fr thomesse@ensem.u-nancy.fr

Abstract

This paper deals with the main role of timing properties in communication stacks (respectively Application processes) to achieve interoperability (respectively interworking) of equipment (respectively Application processes) in real-time distributed applications. The interoperability (respectively interworking) property expresses the global capability of equipment (respectively Application processes) to exchange informations and cooperate in order to meet the end user requirements.

Conformance testing increase consequently the probability of interoperating but does not guarantee it. Experience in communication systems has shown that two equipment, even they succeed to conformance testing, may not be able to interoperate. This is because, first, conformance testing has economic and technical limitations and, second, does not cover performance properties of equipment which are important to achieve temporal interoperability. Hence, an interoperability testing is necessary.

In this paper, we are interested by these performance properties of equipment and their role in obtaining temporal interoperability and interworking.

Keywords Codes: C.2.2; C.2.4; C.2.5
Keywords: Network Protocols; Distributed Systems; Local Networks; Real-time, Interoperability Verification

1 Introduction

This paper deals with the main role of timing properties in communication stacks (respectively Application processes) to achieve interoperability (respectively interworking) of equipment (respectively Application processes) in real-time distributed applications. *The interoperability (respectively interworking) property expresses the global capability of equipment (respectively Application processes) to exchange informations and cooperate in order to meet the end user requirements.*

The main aim of conformance testing is to increase the capability of implementations to be able to work together but tests cannot guarantee conformance to a specification because they allow to detect faults but not their absence. In addition, conformance testing exclude from tests: performance, robustness and reliability evaluation of implementations. Hence, conformance is necessary condition, but not sufficient to guarantee interoperability between conforming implementations. Then, interoperability tests and testing are necessary and recommended [5, 8].

S. T. Vuong et al. (eds.), *Protocol Specification, Testing and Verification XIV*

In relation to conformance testing, interoperability testing try to cover best:
- the interpretation of standards;
- the need of multiple and simultaneous connections;
- the implemented resources capabilities and temporal performances;
- the system heterogeneousness;
- the communication aspect in a real environment.

This paper presents a verification approach of interoperability. Instead of testing interoperability of implementations on a platform, e.g, at the moment of integration, we propose to verify it at the moment of validation of distributed applications. The validation model is built on a net formalism (predicates/transitions net). The model consists on the behaviour of implementations according to their features relating to the interoperability criteria. The validation of interoperability consists on verifying the correctness of interoperability assertions.

The rest of the paper is organised as follows. The second section defines the *profile* and the *layer* interoperabilities and locate them on the framework of OSI model and distributed applications. The third section focuses on the temporal interoperability and its criteria. After a brief description of the Factory Instrumentation Protocol (FIP) fieldbus and its mechanisms, we give an example of a distributed application for illustrating the interoperability relating to temporal performances. The fourth section introduces the verification approach of interoperability and illustrates it with an example. In this section, we describe the model of the distributed application and the kind of interoperability assertions to be verified on it. Finally, the fifth section presents some concluding remarks.

2 Definition of Interoperability and its Criteria

In the area of distributed applications, we distinguish two main components: the material and the functional architectures. The material architecture is the set of equipment. The functional architecture is the set of Application processes. The Application processes are distributed on the equipment and this distribution is called the operational architecture. The behaviour of a distributed application depends on the extent to which its equipment and Application processes are able to cooperate. We qualify the cooperation capability of equipment by *interoperability* and of Application processes by *interworking*. The interoperability of equipment as a whole depends on the peer to peer interoperability of layers in their communication stacks. The location of these definitions on the OSI model is given by figure 1.

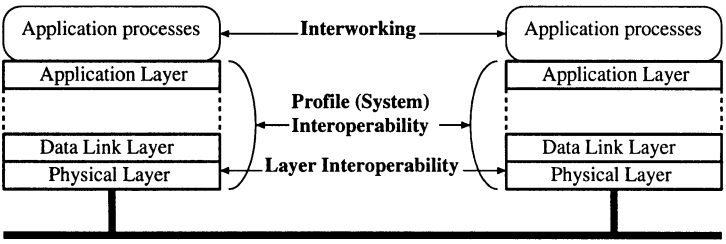

Figure 1: Interoperability and interworking in distributed applications.

We distinguish the interoperability relating to a profile as a whole and the interoperability relating to the layers.

Layer Interoperability

Two or more N-layer entities are called *layer interoperable* if they have passed the protocol conformance testing and it has been shown that they achieve together with success the N-service function like it is specified by the standard and required by the profile.

Profile Interoperability

Two or more communicating entities with the same profile are called *profile interoperable* if their layers are peer to peer *layer interoperable.*

The layer interoperability of protocol conforming layers may vary in accordance with four classes of criteria listed below [1, 2].

Criteria relating to delivered services and to conformance classes

A product implementing Transport class 4 protocol does not interoperate with a product implementing Transport class 0 protocol [6].

Criteria relating to communication protocol between peer to peer layers

Experience in interoperability testing has shown problems relating to difference between protocol or standard versions. For instance, The Draft International Standard (DIS) version of ISO Session layer requires the use of the connection Parameter Group Identifier (PGI) while the International Standard (IS) version of the same standard ISO 8327 makes it optional.

Criteria relating to resources

For example, the maximum number of connections provided by an MMS (Manufacturing Message Specification) implementation or the maximum number of identifier variables provided by a FIP (Factory Instrumentation Protocol) implementation.

Criteria relating to temporal performances

Generally, the time is not specified in the protocols. Consequently, it is not checked in the conformance testing. But, implementations may be not able to interoperate if some delays are incompatible. In a FIP network for example, a refreshment or a promptness delay required in ms (millisecond) unit can not be guaranteed by an implementation which provides only a 10 ms period. And in some protocols, after a message reception, a machine must answer in a given delay.

These classes of criteria are source of non-interoperability. Behaviour of implementations relating to these criteria are not being tested by conformance testing. Hence, it should be tested in interoperability testing. According to the four classes of criteria, we associate four interoperability relations: services interoperability, protocol interoperability, resources interoperability and temporal interoperability. These interoperability relations may be tested separately. In the next section, a sample distributed application is given as an example to illustrate the temporal interoperability and how we validate it.

3 Temporal Interoperability

In this paper, we are interested by performance properties of equipment and their role in obtaining temporal interoperability and interworking. Let us consider an example of a distributed application on the FIP fieldbus. FIP fieldbus network is a communication system of sensors, actuators and control systems which provides mechanisms to specify and guarantee the respect of some temporal constraints. FIP has an architecture composed of three layers: Physical, Data Link and Application layers. The application layer of FIP fieldbus communication system provides several services, for instance, read/write variable value services. The information to be exchanged is based on the broadcast mechanism and the Distributor/Producer/Consumer(s) model (figure 2). The information is produced by the Producer and distributed in a broadcast

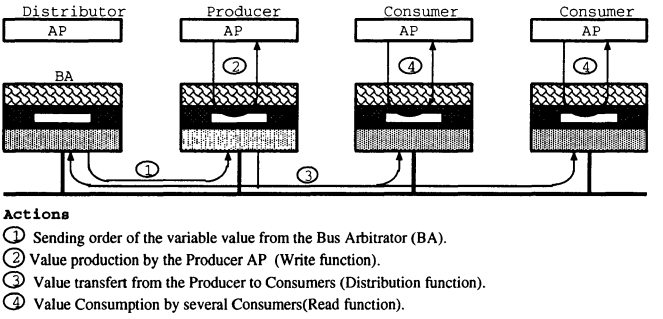

Figure 2: FIP Producer / Distributor / Consumer Model.

manner to all the Consumer(s) in the same time after receiving from the Distributor the sending order of the variable value. FIP offers two kinds of information traffic: periodic and non-periodic variables and messages.

Suppose that the distributed application is composed of one sensor and one actuator. The sensor is responsible for producing the temperature of a process under control (for instance, a machine tool or a nuclear plant reactor, etc) and broadcasting it to the actuator periodically. We suppose here that the actuator plays the roles of a consumer and of the distributor. According to the standard, the distributor broadcasts a production and distribution order $ID\_DAT(var)$, called Protocol Data Unit (PDU), and arms the timer T_0 then waits until receiving the response $RP\_DAT(data)$ or expiring the timer T_0. Suppose the producer needs δ_t units of time to send the variable value on the fieldbus by the $RP\_DAT(data)$ PDU after receiving the sending order. In practice, the value of δ_t for any equipment E_i, called the Equipment Turnaround Time (ETT), is included in an interval of values.

$$\delta_t \; of \; E_i \in [\min(\delta_t) \; of \; E_i, \max(\delta_t) \; of \; E_i]$$

The minimal value $\min(\delta_t)$ is function of equipment hardware characteristics. The maximal value $\max(\delta_t)$ depends on algorithms used for sending the variable value. The two values characterise temporal performances of equipment. In real-time distributed applications "hard" real-time deadlines must often be met for safe operation and system correctness depends not only on the logical result of the system behaviour but also on the time at which the results are produced [4, 7]. As far as temporal interoperability is concerned, it's clear that followed formulae must be satisfied by the sensor and the actuator features:

$\delta_t\ of\ sensor \in [\min(\delta_t)\ of\ sensor, \max(\delta_t)\ of\ sensor]$
$T_0\ of\ actuator \geq \max(\delta_t)\ of\ sensor.$

And in general, to achieve temporal interoperability relating to ETT feature between two equipment, a receiver and a transmitter, the followed formulae must be satisfied [9]:

$$\delta_t\ of\ transmitter \in [\min(\delta_t)\ of\ transmitter, \max(\delta_t)\ of\ transmitter] \quad (1)$$
$$\delta_t\ of\ receiver \in [\min(\delta_t)\ of\ receiver, \max(\delta_t)\ of\ receiver] \quad (2)$$
$$\min(\delta_t)\ of\ transmitter \geq \max(\delta_t)\ of\ receiver \quad (3)$$

The breaking of the above formulae by two communication equipment of a system leads to temporal interoperability problem. Habitually, this kind of problem is discovered at the moment of integration of the distributed application on a platform. This is the way the interoperability testing is achieved [3, 5, 6]. What we propose is to verify the interoperability capability at an earlier stage: at the moment of choosing the equipment of the distributed application. Our approach of interoperability verification is based on measuring out characteristics of each equipment and verification of formulae and logical invariants on a global model of the distributed application and its equipment. The next section gives some details on how to validate the temporal interoperability of an operational architecture of a distributed application according to ETT criterion.

4 Interoperability Verification Approach

The proposed *interoperability verification approach* is based on this idea: first, measurements of characteristics relating to interoperability are done separately on each physical IUT; then, an observation is done on the IUTs global model. The behaviour and features of each IUT relating to an interoperability relation are represented in a model. Then, the interoperability properties, i.e all the criteria of a class which is relating to the interoperability relation to be tested and invoked before, are *verified* on the global model of the IUTs. The model is described by a predicates/transitions system with EVAL predicate/transition analyser [10]. The IUT features relating to the different criteria listed before are either given by the constructor or measured out by a tester and put in a data base. The interoperability model of the IUT is the association of the predicates/transitions net and the data base. The verification of an interoperability relation between IUTs, i.e, the verification of interoperability assertions, is done on the IUTs models. In this way, the interoperability activity seems like the validation one [4, 7] or can be done at the same time. We can say that the proposed interoperability verification is an improvement of the classical validation of distributed applications by considering equipment features relating to interoperability and interworking.

Let us consider the distributed application given earlier and built the predicates / transitions nets of the sensor and the actuator. According to the data link layer standard [9], the behaviour of the sensor and the actuator are represented by the two automatons of figure 3. The behaviour of the distributed application as a whole is the product of actuator and sensor behaviours. The result is the global automaton of figure 4 composed of 4 states and 6 edges. We can distinguish two sub-behaviours. The first sub-behaviour is composed of the states 1 and 2 and the edges $(1, \tau, 2)$ denoted (12) and $(2, \tau, 1)$ (21). The edges 12 and 21 represents the success respectively of the sending order rendezvous and the production response rendezvous between the actuator (the transmitter of the order or the consumer

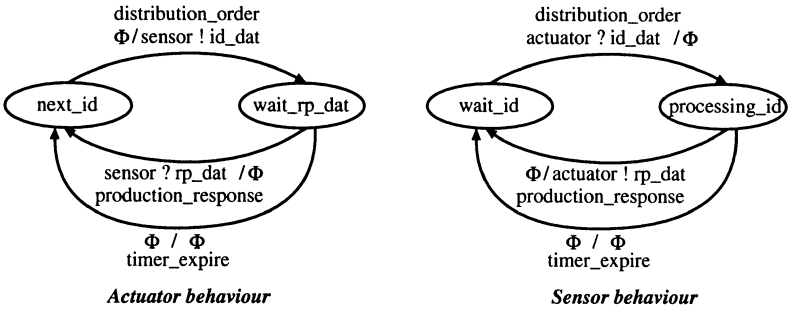

Figure 3: Predicates/transitions nets of the sensor and the actuator.

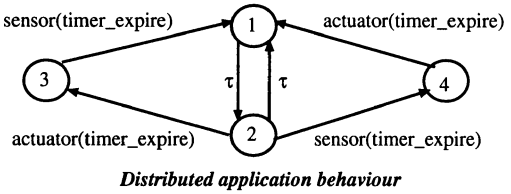

Figure 4: Predicates/transitions net of the distributed application.

of the data) and the sensor (the receiver of the order or the producer of the data). This first sub-behaviour is the one required by the user and expresses the fact that temporal interoperability is satisfied. The second sub-behaviour is composed of the states 1, 2, 3 and 4 and the edges $(1, \tau, 2)$ (12), $(2, actuator(timer\_expire), 3)$ (23), $(3, sensor(timer\_expire), 1)$ (31), $(2, sensor(timer\_expire), 4)$ (24), $(4, actuator(timer\_expire), 1)$ (41). This second sub-behaviour is the one not wished by the user and expresses the fact that temporal interoperability is not satisfied. But at the moment of integration on a platform, the application may have the two sub-behaviours because they are valid. So, the user will detect later the problem of non-interoperability in his application. That's the way it is checked by the interoperability testing approach.

To detect non-interoperability problems earlier, we propose to take into account some features relating to the interoperability criteria at the validation stage of distributed applications. This way, a distributed application will not be declared to be valid if it presents behaviours which are not wished by the user and relating to non-interoperability.

Let us show this idea on the following example. Suppose given the equipment turnaround time values of the actuator and the sensor (instantiation of formulae 1 and 2).

```
/* Min and Max ETT of the Actuator */
minETTactuator(4).
maxETTactuator(7).

/* Min and Max ETT of the Sensor */
minETTsensor(3).
maxETTsensor(5).
```

The interoperability is satisfied if ETT values of actuator and sensor satisfy the following temporal interoperability assertion (see the formula 3) written in Prolog.

```
satisfiedTempIopAssertion :-
                            maxETTsensor(MaxETTreceiver),
                            minETTactuator(MinETTtransmitter),
                            MaxETTreceiver =< MinETTtransmitter.
```

ETT values and the assertion are given in a Prolog data base of the distributed application model.

Non-interoperability case

In order to take into account the ETTs values and the temporal interoperability assertions, we add a clause in some transitions. The clause is *PROVIDED condition*. This clause allows the firing of a transition if and only if *condition* is true. So, *PROVIDED satisfiedTempIopAssertion* and *PROVIDED NOT satisfiedTempIopAssertion* are added to transitions where respectively a rendezvous is done and the timer expire. The global behaviour is computed. The result is 4 states and 5 transitions. There is one less edge then before. That is the edge 21 (see figure 4). If this edge is dead, that means the production response cannot be done by the distributed application because the rendezvous of production is not possible. And if so, that means the equipment will be not able to interoperate on the distributed application platform.

Another way to validate or invalidate the distributed application on the point of view of temporal interoperability is to look for the existence of a path which contains the production response event. So, we compute the path from the initial state 1 to a state where the last edge contains the consumption data transition: actuator(production_response). If the path does not exist, that means the equipment will be unable to interoperate on the distributed application platform in order to exchange the production response data.

Interoperability case

We now suppose the ETT minimal value of the actuator equal to ETT maximal value of the sensor(value 4 replaced by 5). The computation of the global behaviour gives as a result 2 states and 2 edges. There is 4 less edges then before. The four dead edges are 23, 34, 24 and 41 (see figure 4). If these edges are dead, that means the production response can always be done by the distributed application because the rendezvous of production is possible. And if so, that means the equipment will always be able to interoperate on the distributed application platform. We can compute also the path which allow temporal interoperability like before.

5 Conclusion

In this article, first, we have shown that conformance testing is not sufficient to achieve interoperability for two main reasons. The first reason is relating to the consequences of the technical and economic limitations in the test. Some parts of the protocols in conformance tested implementations are not explored. The second reason and the most important that is all the aspects of a real communication system are not considered in conformance testing stages. By studying interoperability problems, we have distinguished several criteria classes: services,

protocol, resources, temporal performances. The interoperability of equipment and problems relating to the interoperability are function of these criteria.

The test of interoperability is based on confrontation of implementations on a platform. We have shown that this verification is done in a later stage of distributed application development.

We have proposed another approach to verify interoperability in earlier stage. This approach is based on the validation of the operational architecture (see section 2) of distributed applications. We suppose that the equipment features relating to the interoperability criteria are given. The validation of the interoperability of distributed applications is done on a model which contains the features and assertions of interoperability. The technique of interoperability validation consists of computing for instance existence of paths or edges which participate to exchange of data between distributed applications equipment.

The main advantage of this approach is to allow the user to validate the interoperability of operational architectures in an earlier stage of the distributed applications design cycle.

References

[1] Y. Benkhellat, M. Siebert, and J.-P. Thomesse. Interoperability of Sensors and Distributed Systems. *Journal Sensors and Actuators A Volumes A37 and A38*, 2:247–254, 1992.

[2] Y. Benkhellat, M. Siebert, and J.-P. Thomesse. Sensors and Distributed Systems Interoperability Criteria. In *Proceedings of EUROSENSORS'93 Conference*, 1993.

[3] J. Gadre, C. Rohrer, C. Summers, and S. Symington. A COS Study of OSI Interoperability. *Computer Standards and Interfaces*, 9:217–237, 1990.

[4] F. Jahanian and A.-L. Mok. A Graph-Theoritic Approach for Timing Analysis and its Implementation. *IEEE Transactions on Computers*, C-36(8):961–975, August 1987.

[5] O. Koné. Deriving Coordinated Testers for Interoperability. In O. Rafiq, editor, *International Workshop on Protocol Test Systems, VI*, pages 335–348. IFIP, University of Pau, Pau, France, 1993.

[6] L. Lenzini and F. Zoccolini. Interoperability tests on OSI products in the framework of the OSIRIDE-Intertest initiative. *Computer Networks and ISDN Systems*, 24:65–79, 1992.

[7] Jonathan S. Ostroff. *Temporal logic for real-time systems.* Research Studies Press LTD, 1989.

[8] O. Rafiq. Le test d'interopérabilité des protocoles. In O. Rafiq, editor, *Ingénierie des protocoles*, pages 543–558, Paris, 1991. Hermès. CFIP'91.

[9] UTE. NF C 46-603: FIP bus for exchange of information between transmitters, actuators and programmable controllers. Data Link Layer. French standard, June 1990.

[10] VERILOG. *EVAL, PREDICATE TRANSITION ANALYSER, RELEASE 1.0.* Toulouse, first edition, October 1991.