IPP-HURRAY! Research Group

Polytechnic Institute of Porto
School of Engineering (ISEP-IPP)

From Simulation to Statistical Analysis: Timeliness Assessment of Ethernet/IP-based Distributed Systems

Nuno PEREIRA
Luís M. Pinho
Eduardo TOVAR

HURRAY-TR-0416 (**version 01**)
April-2004

relatório técnico

technical report

From Simulation to Statistical Analysis: Timeliness Assessment of Ethernet/IP-based Distributed Systems

Nuno PEREIRA, Luís M. Pinho, Eduardo TOVAR

IPP-HURRAY! Research Group
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340502, Fax: +351.22.8340509
E-mail: {npereira, lpinho, emt}@dei.isep.ipp.pt
http://www.hurray.isep.ipp.pt

Abstract:
A number of characteristics are boosting the eagerness of extending Ethernet to also cover factory-floor distributed real-time applications. Full-duplex links, non-blocking and priority-based switching, bandwidth availability, just to mention a few, are characteristics upon which that eagerness is building up. But, will Ethernet technologies really manage to replace traditional Fieldbus networks? Ethernet technology, by itself, does not include features above the lower layers of the OSI communication model. In the past few years, it is particularly significant the considerable amount of work that has been devoted to the timing analysis of Ethernet-based technologies. It happens, however, that the majority of those works are restricted to the analysis of sub-sets of the overall computing and communication system, thus without addressing timeliness at a holistic level.To this end, we are addressing a few inter-linked research topics with the purpose of setting a framework for the development of tools suitable to extract temporal properties of Commercial-Off-The-Shelf (COTS) Ethernet-based factory-floor distributed systems. Thisframework is being applied to a specific COTS technology, Ethernet/IP. In this paper, we reason about the modelling and simulation of Ethernet/IP-based systems, and on the use of statistical analysis techniques to provide usable results. Discrete event simulation models of a distributed system can be a powerful tool for the timeliness evaluation of the overall system, but particular care must be taken with the results provided by traditional statistical analysis techniques.

From Simulation to Statistical Analysis: Timeliness Assessment of Ethernet/IP-based Distributed Systems

Nuno Pereira, Luís M. Pinho, Eduardo Tovar
Polytechnic Institute of Porto
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto, Portugal
{npereira, lpinho, emt}@dei.isep.ipp.pt

Abstract

A number of characteristics are boosting the eagerness of extending Ethernet to also cover factory-floor distributed real-time applications. Full-duplex links, non-blocking and priority-based switching, bandwidth availability, just to mention a few, are characteristics upon which that eagerness is building up. But, will Ethernet technologies really manage to replace traditional Fieldbus networks? Ethernet technology, by itself, does not include features above the lower layers of the OSI communication model. In the past few years, it is particularly significant the considerable amount of work that has been devoted to the timing analysis of Ethernet-based technologies. It happens, however, that the majority of those works are restricted to the analysis of sub-sets of the overall computing and communication system, thus without addressing timeliness at a holistic level.

To this end, we are addressing a few inter-linked research topics with the purpose of setting a framework for the development of tools suitable to extract temporal properties of Commercial-Off-The-Shelf (COTS) Ethernet-based factory-floor distributed systems. This framework is being applied to a specific COTS technology, Ethernet/IP. In this paper, we reason about the modelling and simulation of Ethernet/IP-based systems, and on the use of statistical analysis techniques to provide usable results. Discrete event simulation models of a distributed system can be a powerful tool for the timeliness evaluation of the overall system, but particular care must be taken with the results provided by traditional statistical analysis techniques.

1. Introduction

The factory-floor has been, since a few decades now, one of the major application environments for real-time distributed computing systems [1]. Nevertheless, the evolution of the manufacturing industry has led to the adoption of information technologies, such as the PC and open standards for the factory-floor networking. In fact, the application of information technologies has evolved from relatively passive data collection and reporting roles to feedback control and diagnostics applications. This context unveils the role played by factory-floor networking in modern industrial automation systems.

Interesting, however, is that the use of communication networks at the factory floor is more recent than at the office environment. One of the reasons for this delay was that manufacturing systems usually depend on being able to sample input data at equally spaced points in time [2], and this feature was not easily fulfilled using early office-room networks. This lead to the fieldbus concept, the first step on the road to networked industrial automation systems. In the era of the Internet, however, factory-floor communication systems must also better explore commercial information technologies. This includes Commercial-Off-The-Shelf (COTS) operating systems, TCP/UDP/IP based applications (XML, Java, etc), and general-purpose communication networks such as Ethernet [3, 4].

Nowadays, arguments against the use of Ethernet in industrial environments have almost disappeared. “Familiarity”, “high availability” (subsequently, low cost) and improved timeliness and dependability are driving this phenomenon. But still, there are obstacles to overcome [5]. Indeed, recent research efforts [6, 7] on Ethernet technologies have been focusing on timeliness, trying to find solutions to issues such as bounded response time evaluation, optimal scheduling policies, switching topologies or clock synchronisation. However, they essentially consider the timing characteristics at the Data Link Layer, and it is still to come, to our best knowledge, an overall approach embracing a fully defined protocol stack. While until a couple of years ago a valid justification for this hole could eventually be the actual lack of technologies [8] offering an overall ensemble of protocols and mechanisms, this justification can not serve that purpose anymore.

Timeliness is usually exploited with an underlying framework dominated by the notion of absolute temporal guarantees. For distributed systems, computational and communication loads are presumed to be bounded and known, and the worst-case (at least believed to be) conditions are assumed. In this way, the problem of engineering distributed real-time systems, of which factory-floor distributed computing systems are a representative example, becomes a problem of devising the appropriate tools and methods to assure that all deadlines are met in all circumstances [9].

One option for approaching those tools and methods is through devising the appropriate (analytical) formulations reflecting worst-case conditions. However, and for complex distributed systems, analytical-based worst-case approaches may lead to intractable mathematical models. This further difficult handling and reasoning the analytical abstractions, particularly when techniques such as precedence relationships [10], event phasing [11, 12] or inheritance of time characteristics [13, 14] come to the equation as means of shrinking pessimism levels.

There is another concern that is important to bring into this context. In fact, although the deterministic framework has been proved valid for the deployment of real-time systems in a wide range of applications, it is now accepted that it may pose serious research challenges when trying to apply it to some other application areas. This is eventually the case of some distributed

systems that are more flexible and adaptive in their nature. In this direction, a great amount of research is being performed towards including, into the traditional analytical models for computing worst-case response time, some stochastic representation of the events. Clearly, this may only be good to provide some form of probabilistic guarantees. However, there might be some useful results if the application can cope with occasional deadline misses, within some quantifiable limits [15-20].

Therefore, for the development of tools suitable to extract temporal properties of Ethernet-based factory-floor distributed systems, we advocate the use of a framework that considers several inter-linked research topics.

Firstly, the use of simulation models that mirror the behaviour of the system may allow providing a reasonable framework for the timeliness evaluation of such distributed real-time systems. A modular simulation, in which each node of the system is modelled independently and then combined to form the overall system, enables to easily conceive models of simulation for distinct distributed systems. Different network topologies may be experimented and compared. This approach is being applied to a specific COTS technology, Ethernet/IP [21]. Ethernet/IP, where IP stands for “Industrial Protocol”, is one example of a COTS technology offering a full set of protocols and mechanisms enabling the development of distributed time-critical applications for the factory-floor environment. Ethernet/IP uses an Application protocol – the Control and Information Protocol (CIP), layered on top of a standard TCP/IP protocol stack, where the physical and data link layers can be available Ethernet technologies.

The rest of this paper is structured as follows. The next section presents a brief description of the main components of Ethernet/IP-based distributed systems. In Section 3, we describe how we have been tackling the problem of modelling and simulating distributed systems based on the same COTS technology. In Section 4 we provide a discussion on the use of simulation results to perform statistical timeliness analysis, by means of a concrete simulation example. Finally, in Section 5, we draw some conclusions.

2. Ethernet/IP-based Distributed Systems

Ethernet/IP is a communication system based on encapsulation technologies, suitable for use in industrial environments. It is an open industrial networking standard that takes advantage of commercial, off-the-shelf Ethernet communication chips and physical media. Ethernet/IP defines a protocol stack (Figure 1) that uses an Application protocol named Control and Information Protocol (CIP), layered on top of a standard TCP/IP protocol stack, where the physical medium is Ethernet.

CIP uses an abstract object modelling to describe the suite of communication services available, the externally visible behaviour of a CIP node, and a common means by which information within CIP products is accessed and exchanged [22]. The majority of the messaging

performed on a CIP Network is done through connections. CIP connections define the packets that will be produced on the network. All connections in a CIP network are classified into Explicit Messaging or Implicit Messaging [23]. Explicit Messaging provides generic multi-purpose communication paths between two devices. Implicit messaging provides dedicated, special purpose communication paths between a producing application object and one or more consuming application objects.

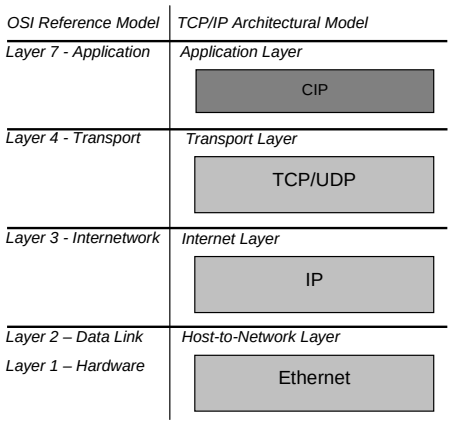

Figure 1. Ethernet/IP Protocol Layers

This last type of connection is used for time-critical I/O data and, therefore, will be the focused type in this paper. Implicit CIP connections may be of four main types: Polled, Change of State, Cyclic, and Strobe. For the purpose of this analysis, the messages passing through the network are assumed to be of Cyclic Implicit CIP connections. Cyclic messages are produced by a device on a predetermined schedule basis, defined by a Requested Packet Interval (RPI) parameter. Any other device that uses the data from the producing device is made aware of the connection identifier associated and accepts any packets from this connection. This producer/consumer model is based upon multicast UDP/IP, which, in turn is mapped over multicast Ethernet/IP. There is a one-to-one correspondence between a producer/consumer connection and a multicast group.

Ethernet/IP Networks are constituted of three basic elements: Remote I/Os, Controllers and interconnecting Switches. These elements communicate with each other via Ethernet. The Remote I/O and Controller nodes can be composed by a number of different modules communicating via a device-specific backplane (Figure 2). Typically, a Controller is composed of a number of I/O modules (labelled in the figure as *I* or *O*), several Controller modules (*C*) and one or more Ethernet Adapters (*EA*). A Remote I/O is similar to a Controller, but it is simpler as it lacks Controller modules.

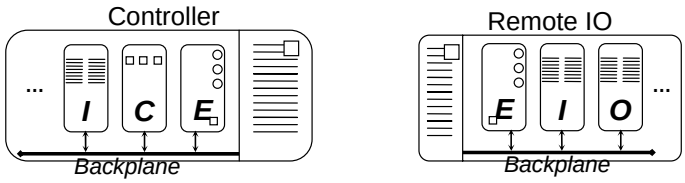

Figure 2. (*Controller* and *Remote IO*) Ethernet/IP basic nodes

Before closing this section it is worth to take a closer look to the type of end-to-end transactions we are addressing. Assume a simple network scenario (Figure 3), and an end-to-end transaction between the Remote I/O and the Controller nodes.

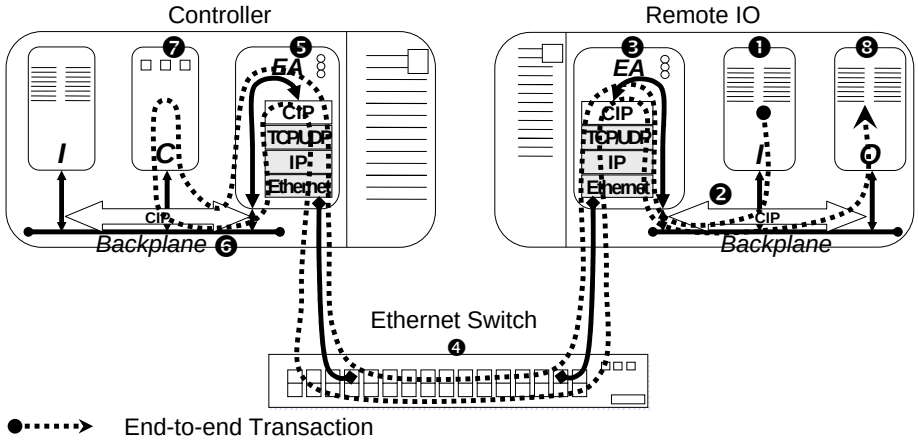

Figure 3. End-to-end transaction path example

The above mentioned transaction starts at the input module of the Remote I/O (①), where a message with the actual input data will be generated (produced) at a rate defined by the RPI parameter for that particular connection. This message will suffer contention delay at the node device backplane (②), and will then arrive at the Ethernet Adapter, where it is processed, and sent via the Ethernet communication interface (③). The Ethernet switch forwards the message to the corresponding output port(s) (④) with a determinable latency. The message will arrive to the Controller Ethernet Adapter (⑤), where is parsed and dispatched to the Controller module via the node backplane (⑥). At the controller (consumer of the data related to the transaction), the input data will be processed by a controller task (characterised by a worst-case response time), that generates the corresponding output data (⑦). The generated output data corresponds to another transaction, in this case produced by the controller and consumed at the Remote I/O node. With another RPI parameter associated, this message will then follow the inverse path (⑥,⑤,④), until it reaching the EA of the Remote I/O (③). It is then processed and delivered to the output module that will, in result, energise the corresponding output(s) (⑧).

The producer/consumer model of Ethernet/IP-based systems, together with the advantages of Ethernet technologies in the factory-floor greatly increase the interest put on using this technology for time-dependent distributed applications.

3. Ethernet/IP Simulation Model

The Ethernet/IP distributed system simulation environment was developed using the OMNeT++ [24] discrete event simulation platform. OMNeT++ is an object oriented modular discrete event simulator, which provides a reusable component framework, where the system components can be independently built and then characterised and assembled into larger components and models. The basic system components are built using the C++ language and then assembled into larger components and models using a high level language, named NED (a OMNeT++ specific scripting language). An OMNeT++ model consists of hierarchically nested Modules. These modules can have parameters which are used to customise the module behaviour; to create flexible model topologies; and for module communication, as shared variables. Modules can also communicate through message passing, where messages can contain arbitrary data structures.

Our simulation model for Ethernet/IP is composed of three basic components (nodes), mapping on the main Ethernet/IP devices: a Remote IO, a Controller and an Ethernet Switch. Each of these basic nodes can be instantiated into several different device models, with different particular characteristics, since modularity and parameterisation are considered into the design to a sufficient extent.

Figure 4 depicts a basic example of a network model. This network is composed of a Remote IO, a Controller and a Switch. In the next subsections, further details are provided concerning model implementation aspects.

3.1. Remote IO Node

The Remote IO is composed of several IO modules and an Ethernet/IP Adapter. The IO modules contain the several input/output connections of the device. Typically, each IO module will act has an Input or Output module, but not both at the same time. As previously mentioned in Section 2, the diverse modules inside a Remote IO node communicate (CIP packets) through a backplane. The Ethernet Adapter is responsible for relaying messages between the Backplane and the Ethernet network. CIP packets are eventually (for the case of a consumer outside the node) encapsulated into UDP packets inside the Ethernet/IP Adapter (*ethIPAdapter* in Figure 4).

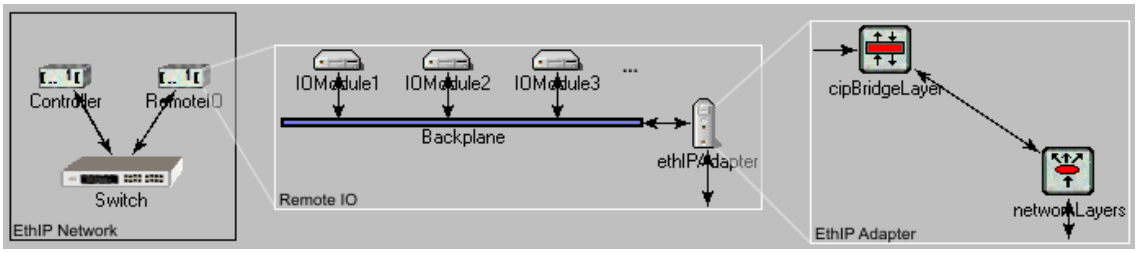

Figure 4. OMNET++ hierarchical models

The Backplane is a simulation module that exists both at Controller and Remote IO nodes. It gathers information about the data connections produced/consumed at each module. Figure 5 provides a sample of NED code defining the Backplane OMNeT++ simple module. A simple OMNeT++ module is declared with the keyword *simple*, followed by the module's name. The declaration of the simple module terminates with the keyword *endsimple*. Included in the declaration are the OMNeT++ simple module's parameters and gates. The gates of an OMNeT++ module define the entry points of the module. For the example of the Backplane module, an array of input and output gates are defined, where each pair of input and output gates represents a Backplane interface, connecting to a node's module.

```
simple Backplane
    parameters:
            tTableTime : numeric,
            frameTime  : numeric,
            timeDivison: bool;
    gates:
        in: in[];
        out: out[];
endsimple
```

Figure 5. Backplane OMNET++ simple module NED definition

The Backplane simple module has the parameter *tTableTime*, which defines the transmit table time, used for the time division multiple access (TDMA) protocol used as backplane's medium access control protocol. The parameter *frameTime* concerns the time a message takes to be transmitted in the Backplane, and the parameter *timeDivision* specifies whether the time division protocol behaviour should be precisely simulated or simplified. The Backplane module simulates the behaviour of a TDMA contention schema where access to the communication medium is equally distributed to the several producing connections delivering data to the Backplane. Nevertheless, and because this simulation approach of the Backplane can introduce a great amount of events, it is possible to disable this behaviour. The alternative will then be to insert a variable delay, as a function of the number of connections that send messages to the Backplane.

The Ethernet/IP Adapter is responsible for relaying messages to/from the Ethernet network. It does its job by receiving the CIP messages from the Backplane and, in the CIP Bridge Layer

(*cipBridgeLayer* in Figure 4), encapsulating them into UDP packets which are passed down to the Network Layer of the UDP/IP stack. CIP packets are encapsulated into IP packets and an Ethernet frame is created and sent through the network. On the opposite direction the packets are retrieved of the UDP/IP packet and delivered to the Backplane.

The Ethernet/IP Adapter models the delays introduced to perform the encapsulation of the messages, to access the network and the concurrent access to the adapter resources. Figure 6 illustrates the NED definition of the Ethernet Adapter OMNeT++ module (a compound OMNeT++ module). A compound module is composed of other modules. It is declared with the keyword *module* and the module's name, closed by *endmodule*. Like an OMNeT++ simple module, a compound module is composed of the module's parameters and gates. Additionally, it has to include its sub-modules and the connections between the sub-modules and gates.

```
module EthIPAdapter
    parameters:
        connectionIDProducedList : string,
        connectionIDConsumedList : string;
    gates:
        in: from_backplane;
        out: to_backplane;
        in: from_eth;
        out: to_eth;
    submodules:
        cipBridgeLayer: CIPBridgeLayer;
        networkLayers: NetworkLayers;
    connections:
        from_backplane --> cipBridgeLayer.from_bp[0];
        to_backplane <-- cipBridgeLayer.to_bp[0];
        networkLayers.to_application --> cipBridgeLayer.from_ntw;
        networkLayers.from_application <-- cipBridgeLayer.to_ntw;
        from_eth --> networkLayers.from_phy;
        to_eth <-- networkLayers.to_phy;
endmodule
```

Figure 6. *EthIPAdapter* OMNET++ compound module NED definition

Two parameters are considered. The *connectionIDProducedList* parameter and the *connectionIDConsumedList* parameter, for listing the CIP connection identifiers of the connections produced and consumed in the node's modules connected to the Backplane, respectively. The sub-modules considered into an *EthIPAdapter* module are the CIP Bridge Layer (*cipBridgeLayer* sub-module) and Network Layer (*networkLayers* sub-module). The connections implemented (refer to the NED code sample in Figure 6) are those between these two layers and the input/output gates from both the Backplane and the Ethernet network.

Each of the IO modules (labelled *IOModule1, IOModule2 and IOModule3* in Figure 4) inside a node and connected to the Backplane has a CIP Layer, responsible for delivering data to/from the IO connections. The IO Connection can behave either as an output or input connection, and each IO Module may have several input or output connections connected to its CIP Layer (Figure 7).

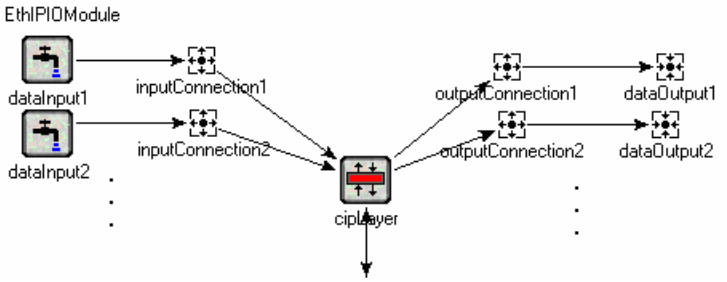

Figure 7. ONNET++ *EthIPIOModule* composition

When an IO Connection is doing the task of an input connection, it receives data from a data input, which generates input data at a defined periodicity (this data input models the input signals of an input connection). At a defined Requested Packet Interval (RPI), the IO Connection constructs a CIP data item from the last received data, and sends it to the CIP Layer. When an IO Connection is acting like an output connection, it receives data from the CIP Layer, which is delivered to a data output, after a parameterised hardware delay. This is illustrated in Figure 8, which provides the C++ code of the message handler from the *IOConnection* class.

```
void IOConnection::handleMessage(cMessage *msg) {
 if (msg->isSelfMessage() == true && inputModule == true) {
  // at rpi, send input data and schedule next rpi
  sendInputData();
  if (((simtime_t)*rpi) > 0)
   scheduleAt(simTime()+((simtime_t)*rpi), msg);
  } else {
   if (inputModule == true) { // acting as an input
    // discard previous dataItem and store new one
    if (dataItem != NULL) delete dataItem;
     dataItem = (CIPDataItem*) msg->dup();
    } else // acting as an output
    sendDelayed(msg->decapsulate(),((simtime_t)*asicDelay), "out");
    delete msg; // After finishing with a message, it is released
  }
}
```

Figure 8. C++ code for the *IOConnection* class message handler

The data input generators (*dataInput1*, *dataInput2*, ..., in Figure 7) model the signals applied at the input pins of the IO. They are characterised by two delays introduced after the generation of the input (a hardware delay and a filter delay). They are also characterised by the length of the data generated and by the periodicity of the data generation. OMNet++ supports defining any of these parameters as a randomly distributed function, with characteristics defined by the user. These parameters can be either defined in the NED code of a compound module, in which case it will be the same for all instances of this compound module, or defined in a special initialisation file that may assign the parameters of each module in the simulation.

Figure 9 exemplifies the definition of the *dataInput* (NED code) parameters in an IO module. Note that the generation period of the *dataInput* is defined as a random variable. In the following, it is defined with a uniform distribution in the interval [100, 150] milliseconds.

```
module EthIPIOModule
…
    submodules:
        dataInput: Input[numInputs];
            parameters:
                hwDelay = 200 us,
                dataLength = 22,
                filterDelay = 0,
                period = uniform (0.1, 0.15);
…
endmodule
```

Figure 9. OMNET++ *EthIPIOModule* NED code for parameter configuration

Figure 10 illustrates the alternative of setting the same parameters through an initialisation file, for a particular IO module instantiation (*ioModule1*), inside of a Remote IO node (*ethIPIO1*), within a network (*ethIPNetwork1*).

```
ethIPNetwork1.ethIPIO1.ioModule1.dataInput[0].hwDelay = 200 us
ethIPNetwork1.ethIPIO1.ioModule1.dataInput[0].dataLength = 22
ethIPNetwork1.ethIPIO1.ioModule1.dataInput[0].filterDelay = 0 ms
ethIPIO1.ioModule1.dataInput[0].period = uniform(0.1,0.15)
```

Figure 10. *EthIPIOModule* parameter configuration through initialisation file

3.2. The Controller Node

The Controller node is, in its structure, similar to the Remote IO node. The Backplane, the Ethernet/IP Adapter and IO Modules are exactly the same modules as described previously for the Remote IP node (Section 3.1). Of course, it is possible to parameterise each of the modules differently, and therefore manipulate their actual behaviour.

There is however a module that must be specified for the particular case of Controller Nodes: the Controller module. In the actual Ethernet/IP systems, the Controller module is used to perform control functions. The controller was modelled reusing some OMNeT++ modules described earlier. It reuses the IO Connection modules and the CIP Layer, seen previously in the IO Module. In this case, the naming used for these modules may be somewhat misleading. Output connections refer to actual input data (from the input device) and output connections refer to actual output data (from the controller to the output IO). Thus, in the Controller, the data received from output connections is processed by a controller task which, after an upper-bounded response time will generate the respective data, to be delivered to input connections.

Currently, the controller only possesses one task that processes all the output data received, in order to simplify the construction of simulation networks. The modelling details for Controller modules are presented in Figure 11.

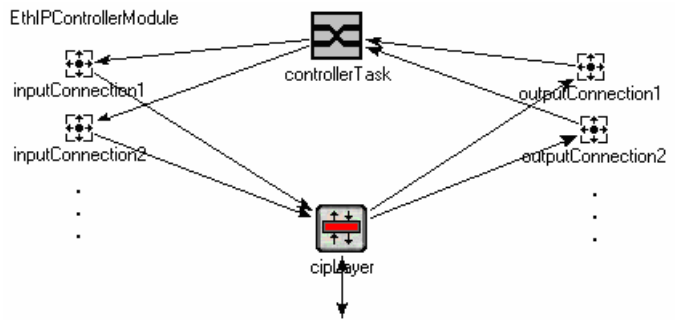

Figure 11. ONNET++ Controller module composition

In the current implementation, the correspondence between the output data received and the input data generated by the Controller task is made in a one to one relation between the indexes of the connections. This means that, at reception of data from the output connection with index one, the task will generate an input also at index one after the respective task response time. The C++ message handler that reflects this simple behaviour is illustrated in Figure 12.

```
void ControllerTask::handleMessage(cMessage *msg)
{
  sendDelayed(msg, (simtime_t)*responseTime, "out",
              msg->arrivalGate()->index());
}
```

Figure 12. C++ code for the Controller Task class message handler

The task (worst-case) response time is a Controller Task parameter which will be a time span introduced between the receptions of data until the generation of the data to be delivered. Again, this parameter can be defined has a random function.

3.3. The Switch Node

The Switch node models the delays introduced by an Ethernet Switching component. For the purpose of this simulation, it is only necessary that the Switch recognises multicast groups and deliver the frames received in an appropriate manner. The Switch model is composed of several ports that connect to the nodes in the network. Because there is a port in each direction, the Ethernet medium is assumed to be full-duplex.

The Switch node is a simple OMNeT++ module. The NED definition of the Switch OMNeT++ module is rather simple, and is given in Figure 13. It is similar to the Backplane

OMNeT++ module, since it has an array of input and output gates, in which each pair represents the interface with each connecting modules (the switch port).

```
simple Switch
    parameters:
        nodename : string,
        switchDelay : numeric;
    gates:
        in: in[];
        out: out[];
endsimple
```

Figure 13. OMNET++ Ethernet Switch simple module NED definition

OMNeT++ offers a rather convenient manner of defining channel transmission characteristics. It is possible to define the characteristics of the connection between any two modules by using a predefined *channel*. A channel is defined with its name, preceded by the keyword *channel*. A channel may be assigned with the attributes *delay*, *error* and *datarate*. The example code depicted in Figure 14 corresponds to the definition of a 100 Mbit/sec Ethernet channel with a normally distributed delay, with mean value of 150 μs and a standard deviation of 50 μs. The connecting channels model the transmission delays and queue the messages whenever concurrent access to the medium occurs.

```
channel ethernet
    delay normal(0.00015,0.00005);
    datarate 100*10^6;
endchannel
```

Figure 14. Ethernet Channel definition in OMNET++

To simplify the multicast deliver process, the connection identifier of a producing connection is directly mapped into the last octet of an IP Multicast Address. For example, for a connection with the identifier 128, the IP Multicast Address would be constructed with a user defined prefix and the last octet being 128; that is, for a prefix of 239.0.0., the connection with identifier 128 would be mapped to the multicast group with address 239.0.0.128.

Because mapping rules defined by multicast Ethernet MAC address mapping are also used [25], the Ethernet frames actually contain the connection identifier mapped into the multicast groups. In this way, it is possible for the Switch to simply construct, at initialisation time, a list of all producing/consuming connection IDs for each connected node. At run time, the Switch module will merely compare the connection identifiers of the received frames with the ones in the list for each node, swiftly delivering copies of the received frame to all nodes that belong to the multicast group.

The Switch is parameterised by a delay that represents the time taken to process the frames, which can also be defined as a random function.

4. Discussion of Results over a Practical Example

In order to provide some insight into the obtainable results with this modelling and simulation approach for Ethernet/IP-based distributed systems, an example system is presented. The results of its simulation and how they could be correctly analysed are then discussed in this section. Note that we are aiming at obtaining an estimation of the end-to-end response time for a number of transactions. A primary goal is to disclose some fundamental aspects about the analysis of the simulation results.

4.1. Example scenario

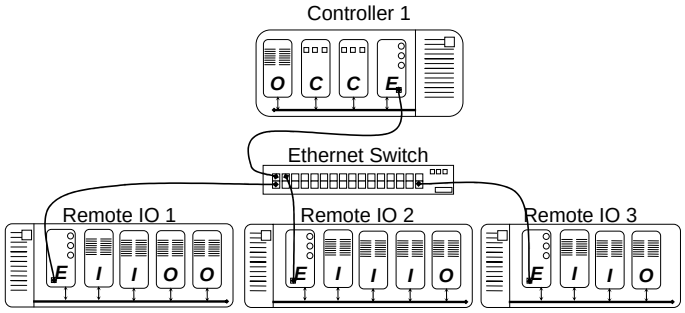

Figure 15. Example of simulated system

The example system is constituted of three Remote IOs, one Controller and an interconnecting switch (Figure 15).

The Controller node is composed of one IO module and two Controller modules. The first Remote IO includes four IO modules, two for output and two for input. The second Remote IO also includes four IO modules, three for input and one output. Finally, the last Remote IO contains three IO modules, two for input and one output.

The system has nine end-to-end transactions between the Remote IOs and the Controller. This results in a total of eighteen connections, half from the Remote Input to the Controller (*Input direction*) and the other half, from the Controller to the Remote IO (*Output direction*).

Table 1 presents the connection identifiers attributed to the connections that make up the several transactions, whereas, Table 2 provides further details on these connections.

Table 1. End-to-end transactions in example network

Transaction	*Connection at Input Direction*	*Connection at Output Direction*
1	131	141
2	132	142
3	133	143
4	134	144
5	151	161
6	152	162
7	153	163
8	171	181
9	172	182

Table 2. Connections in example network

Node	*Module*	*Input Connection*		*Output Connection*
		ID	*RPI (ms)*	*ID*
Controller	*Controller module 1*	141	10	131
		142	7	132
		143	25	133
		163	35	153
	Controller module 2	144	20	134
		161	350	151
		162	110	152
		181	200	171
		182	350	172
Remote IO 1	*IO module 1*			141
				142
				143
				162
	IO module 2			163
	IO module 3	171	200	
		172	350	
Remote IO 2	*IO module1*	131	10	
	IO module2	132	7	
	IO module 3	133	25	
		134	20	
	IO module 4			144
				163
Remote IO 3	*IO module 1*	151	55	
		152	80	
		153	75	
	IO module 2			161
				181
				182

4.2. Analysis of simulation output data

It is known that not much can be concluded with a single simulation run. In fact, the results of a given simulation run are just particular realisations of random variables that may have large variances. It is also known that classical statistical techniques based on Independent and Identically Distributed (IID) observations are not directly applicable to the investigation of simulation results. In fact, simulation output data results are usually highly correlated and have non-stationary distributions.

Several different methods have been developed to correctly compute estimates of a model's characteristics [26]. There is however no simple or complete solution. Besides, the precision of the estimation is at the cost of long and computing intensive simulation runs. Although previous works have interesting approaches for the application of discrete–event simulation to the analysis of distributed real-time systems (e.g. [27]), to our best knowledge, little has been advanced in respect to the actual statistic analysis of the simulation output results, including some measure of confidence in the results.

Most of the methods for the analysis of simulation output data, referred in the literature [26], rely on the fact that, although the simulation results of a single simulation run are not independent, it is possible of obtain independent observations across the results of several independent simulation runs (replications). A set of replications is independent if the random numbers used to drive the simulation through time are different for each replication.

Mainly, there are two general strategies that can be efficiently approached. One relates to fixed-sample-size procedures, where a single simulation run of a fixed length is made, and then one of a number of procedures is used to construct a confidence interval from the available data. A second can be based on sequential procedures that increase sequentially the length of a single simulation run until an acceptable confidence interval can be constructed.

A particular method, included within the group of fixed sample procedures, is the *replication/deletion*. It is a fairly simple method, with a reasonably good statistical performance [26], which we will briefly describe and apply in the analysis of the simulation network example presented formerly. The goal is to obtain an estimate and confidence interval for a steady-state mean v of worst-case observations.

Suppose that we make n replications of the simulation each of length m, where m is much larger than l (the warm-up period used to eliminate the initial transient problem). Let X_i be independent and identically distributed (IID) random variables given from the maximum end-to-end response time observed in each simulation replication i, in the set of response times between l and m. X_i holds an expected average approximate of the steady-state mean v, across i replications of the simulation. Thus, $\overline{X}(n)$ is an approximately unbiased point estimation for v, and an approximate $100(1 - \alpha)$ percent confidence interval for v may be obtained by [26]:

$$\overline{X}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{n}} \quad \textbf{(1)}$$

where $\overline{X}(n)$ is:

$$\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n} \quad \textbf{(2)}$$

and $S^2(n)$ is computed using the following equation:

$$S^2(n) = \frac{\sum_{i=1}^{n} \left[X_i - \overline{X}(n)\right]^2}{n-1} \quad \textbf{(3)}$$

The half-length of the replication/deletion confidence interval given by equation (1) depends on the variance of X_j, which will be unknown for the first n replications. Therefore, it is necessary to make a sufficient number of replications of the simulation to achieve a confidence interval small enough for a particular purpose.

4.3. Statistical results of the simulation

Table 3 provides the results of the application of such approach to the analysis of the simulation output data. In this, we will attempt to construct a confidence interval for the worst-case that can be expected in the long run. This estimation is based on the observation of successive maximum end-to-end values verified across simulation replications and the variance of these observations. The number of replications performed was 61, which was a number of replications that allowed obtaining an error below 25-26% of the estimate for all transactions.

Table 3. Results of simulation output using replication/deletion

Transaction	*Estimation for 99.9% confidence interval ($X \pm \varepsilon$ ms)*	*99.9% Confidence interval (ms)*
Tr. 1	21.22 ± 4.42	[16.80 , 25.64]
Tr. 2	15.28 ± 3.97	[11.31 , 19.26]
Tr. 3	51.15 ± 5.08	[46.07 , 56.24]
Tr. 4	41.11 ± 4.68	[36.43 , 45.78]
Tr. 5	700.45 ± 9.22	[691.23 , 709.66]
Tr. 6	220.90 ± 6.27	[214.62 , 227.17]
Tr. 7	110.20 ±12.79	[97.41 , 122.98]
Tr. 8	400.74 ± 7.44	[393.30 , 408.18]
Tr. 9	700.59 ± 8.72	[691.87 , 709.31]

The X presented in table above represents the estimation for the worst-case response time of the transactions. The margin of error (ε) gives a measure on how accurate the estimation is,

based on the variability of the estimation. The confidence level (99.9%) reflects the amount of confidence that, in the long run, this approach will be able to approximate the true worst-case. With these values, it is possible to construct the confidence intervals displayed.

This type of conclusion about the behaviour of a concrete system may be of relevance to the systems designer, when a probabilistic evaluation of the system is being carried out. Nonetheless, some relevant remarks that might be raised towards this analysis include the fact that the simulation data needed to produce such results may be at a prohibitive computation cost. This time actually depends on a number of variables. The complexity of the system influences the number of events generated during the simulation, the variance of the variables under study affect the size needed for each individual simulation replication, and the margin of error desired, which is also influenced by the variation of the variables of interest, may be controlled by the number of simulation replications. A close investigation of these matters is beyond the scope of this paper, but this is an important issue that must be evaluated in order for this approach to succeed. Nevertheless, it can be advanced that, for the example presented, each replication took less than 2 minutes to run on a fairly old machine (PIII 1GHz).

Also, as noted, the precision obtained depends on the variance of the variables. There are methods to reduce the variance of a simulation output, which generally require controlling random-number streams to introduce correlation in successive observations. Such methods are usually dependent on a particular model and, if not carefully used may impair the validity of the results. Nonetheless, regardless of such techniques, by observing the evolution of the data obtained it is clear that there is a level of precision which can not be much improved by increasing the number of simulation replications. Therefore, particular care must be taken with the use of traditional statistical methods when timeliness guarantees must be provided.

5. Summary and Conclusions

Ethernet-based technologies have already gained a strong position in the factory-floor. For many years, deemed non-determinist, Ethernet has gone through some evolution which enables its use in real-time applications. Nevertheless, Ethernet technology, by itself, does not include features above the lower layers of the OSI communication model. Although lots of attention has been devoted to the timing analysis of Ethernet-like technologies and solutions, most of the work on Ethernet has been restricted to the Data Link Layer level. It is still to come an overall approach that allows the evaluation of a whole Ethernet-based distributed computing system.

In this paper, we have presented the modelling and simulation of Ethernet/IP-based systems, which is being addressed with the purpose of setting up a framework for the development of tools suitable to extract temporal properties of Commercial-Off-The-Shelf (COTS) Ethernet-based factory-floor distributed systems as a whole. The use of discrete event simulation models can be a powerful tool for the timeliness evaluation of the overall system, but particular care

must be taken with the results provided by traditional statistical analysis techniques. Therefore, some discussion was also introduced on the use of simulation results to perform statistical timeliness analysis. This discussion provides insights for ongoing work in this area. In order to obtain better more approximate estimates to real-time system parameters, analysis techniques that incorporate some of the features of such systems must be considered.

References

[1] E. Tovar, "Supporting Real-Time Communications with Standard Factory-Floor Networks", PhD Thesis, University of Porto, Porto, 1999. Available online at http://www.hurray.isep.ipp.pt/asp/show_doc.asp?id=60.

[2] T. Skeie, S. Johannessen, and O. Holmeide, "The Road to and End-to-End Deterministic Ethernet", in proceedings of the 4th IEEE International on Factory Communication Systems (WFCS'2002), Vasteräs, Sweden, pp. 3-9, 2002.

[3] Rockwell Automation, "Making Sense of e-Manufacturing: a Roadmap for Manufacturers", Rockwell Automation, White Paper 2000. Available online at http://www.me.umist.ac.uk/staffpgs/awl/Rockwell%20emfg%20White%20Paper.pdf.

[4] M. Alves, E. Tovar, G. Fohler, and G. Buttazzo, "CIDER: Envisaging a COTS Communication Infrastructure for Evolutionary Dependable Real-Time Systems", in proceedings of the WIP Session of the 12th IEEE Euromicro Conference on Real-Time Systems, Stockholm, Sweden, pp. 19-22, 2000.

[5] Berta Baptista, "Industrial Ethernet: Building Blocks for a Holistic Approach", in proceedings of the WIP Session of the 4th IEEE International Workshop on Factory Communication Systems (WFCS'02), Vasteras, Sweden, 2002.

[6] Yequiong Song, Anis Koubaa, and Francois Simonot, "Switched Ethernet for Real-Time Industrial Communication: Modelling and Message Buffering Delay Evaluation", in proceedings of the 4th IEEE Int. Workshop on Factory Communication Systems (WFCS02), Vasteräs, Sweden, pp. 27-35, 2002.

[7] Jean-Philippe Georges, Eric Rondeau, and Thierry Divoux, "Evaluation of switched Ethernet in an industrial context by using the Network Calculus", in proceedings of the 4th IEEE Int. Workshop on Factory Communication Systems (WFCS02), Vasteräs, Sweden, pp. 19-26, 2002.

[8] M. Volz, "Quo Vadis Layer 7",in The Industrial Ethernet Book, vol. no. 5: Spring, 2001, pp. 8-10.

[9] G. Buttazzo, "Hard real-time computing systems predictable scheduling algorithms and applications": Kluwer Academic Publishers, 1997. Web Site: http://www.netLibrary.com/urlapi.asp?action=summary&v=1&bookid=39600.

[10] J. C. Palencia Gutierrez and Michael Gonzalez Harbour, "Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems", in proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99), Phoenix, Arizona, pp. 328-339, 1999.

[11] Ken Tindell, "Adding time-offsets to schedulability analysis", University of York Dept. of Computer Science, Heslington, York, England, Technical Report YCS-94-221, 1994. Available online at http://www.cs.york.ac.uk/rts/papers/techrep.html.

[12] J. C. Palencia Gutierrez and Michael Gonzalez Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets", in proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS'98), Madrid, Spain, pp. 26-37, 1998.

[13] Ken Tindell, "Holistic schedulability analysis for distributed hard real-time systems," Microprocessors and Microprogramming, vol. 50, pp. 117-134, 1994.

[14] Marco Spuri, "Analysis of Deadline Scheduled Real-Time Systems", INRIA, Technical Report 2772, April 1996. Available online at http://www.inria.fr/rrrt/rr-2772.html.

[15] G. Buttazzo and L. Abeni, "QoS Guarantee using Probabilistic Deadlines", in proceedings of the IEEE Euromicro Conference on Real-Time Systems (ECRTS'99), York, UK, pp. 242-249, 1999.

[16] J. Díaz, D. García, K. Kim, C. Lee, L. Lo Bello, J. López, S. L. Min, and O. Mirabella, "Stochastic Analysis of Periodic Real-Time Systems", in proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02), Austin, Texas, pp. 289-300, 2002.

[17] M. K. Gardner, "Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems", PhD thesis, University of Illinois, Urbana Champaign, 1999. Available online at http://www.cs.uiuc.edu/Dienst/UI/2.0/Describe/ncstrl.uiuc_cs/UIUCDCS-R-99-2114.

[18] John P. Lehoczky, "Real-Time Queueing Network Theory", in proceedings of the 18th The IEEE Real-Time Systems Symposium (RTSS'97), San Francisco, California, pp. 58-67, 1997.

[19] John P. Lehoczky, "Real-Time Queueing Theory", in proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS'96), Washington, DC, pp. 186-195, 1996.

[20] T. S. Tia, Z. Dens, M. Shankar, M. Storch, J. Sun, L. C. Wu, and J. S. Liu, "Probabilistic Performance Guarantees for Real-Time Tasks with Varying Computation Times", in proceedings of the RealTime Technology and Applications Symposium (RTAS'95), Chicago, Illinois, pp. 164-173, 1995.

[21] "Ethernet/IP Specification, Release 1.0", vol. 1-2: ControlNet International and Open DeviceNet Vendor Association, June, 5th, 2001.

[22] V. Schiffer, "The CIP Family of Fieldbus Protocols ans its newest Member - Ethernet/IP", in proceedings of the 8th IEEE International conference on Emerging Technologies and Factory Automation (ETFA'01), Antibes - Juan les Pins, France, pp. 377-384, 2001.

[23] Paul Brooks, "Ethernet/IP - Industrial Protocol", in proceedings of the 8th IEEE International conference on Emerging Technologies and Factory Automation (ETFA'01), Antibes - Juan les Pins, France, pp. 505-514, 2001.

[24] A. Varga, "OMNeT++ Discrete Event Simulation System", v2.3, 2004. Web Site: http://www.omnetpp.org/.

[25] S. Deering, "Host Extensions for IP Multicasting", Stanford University RFC 1112, August 1989. Available online at http://www.ietf.org/rfc.html.

[26] Averill M. Law and W. David Kelton, Simulation modelling and analysis, 3rd ed. New York: McGraw-Hill, 2000.

[27] A. Wall, J. Andersson, and C. Norström, "Probabilistic Simulation-based Analysis of Complex Real-Times Systems", in proceedings of the 6th IEEE International Symposium on Object-Oriented Real-time distributed Computing (ISORC'03), Hakodate, Hokkaido, Japan, pp. 257-268, 2003.