IPP-HURRAY! Research Group

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

INDEPTH: Timeliness Assessment of Ethernet/IP-based Systems

Nuno PEREIRA
Eduardo TOVAR
Luis Miguel PINHO

HURRAY-TR-0433
October-2004

relatório técnico

technical report

INDEPTH: Timeliness Assessment of Ethernet/IP-based Systems

Nuno PEREIRA, Eduardo TOVAR, Luis Miguel PINHO

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, emt, lpinho}@dei.isep.ipp.pt
http://www.hurray.isep.ipp.pt

Abstract:
The continuous improvement of Ethernet technologies is boosting the eagerness of extending their use to also cover factory-floor distributed real time applications. Indeed, it is remarkable the considerable amount of research work that has been devoted to the timing analysis of Ethernet-based technologies in the past few years. 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 in a holistic fashion. To this end, in this paper we address an approach, based on simulation, aiming at extracting 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 modeling and simulation of Ethernet/IP-based systems, and on the use of statistical analysis techniques to provide useful results on timeliness.

INDEPTH: Timeliness Assessment of Ethernet/IP-based Systems

Nuno Pereira, Eduardo Tovar, Luís Miguel Pinho
IPP-HURRAY Research Group, Polytechnic Institute of Porto
P-4200-072 Porto, Portugal
{npereira, emt, lpinho}@dei.isep.ipp.pt

Abstract

The continuous improvement of Ethernet technologies is boosting the eagerness of extending their use to also cover factory-floor distributed real time applications. Indeed, it is remarkable the considerable amount of research work that has been devoted to the timing analysis of Ethernet-based technologies in the past few years. 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 in a holistic fashion. To this end, in this paper we address an approach, based on simulation, aiming at extracting 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 modeling and simulation of Ethernet/IP-based systems, and on the use of statistical analysis techniques to provide useful results on timeliness.

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, 2]. 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 [3], and this feature was not easily fulfilled using early office-room networks.

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 [4]. Additionally, and in the era of the Internet, factory-floor communication systems must also better explore commercial information technologies [5]. This should include TCP/IP-based applications (XML, Java, etc.) and general-purpose communication networks such as Ethernet, just to mention a few example technologies.

Regarding Ethernet technologies and, more importantly, distributed systems based upon them, guaranteeing timeliness is still, most times, an open issue. In fact, the majority of 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 synchronization. However, they essentially consider the timing characteristics at the Data Link Layer. 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 gap could eventually be the lack of technologies offering an overall ensemble of protocols and mechanisms [8], this justification can not serve that purpose anymore. In fact, there are already Commercial-Off-The-Shelf (COTS) solutions for Ethernet-based systems providing a fully defined communication protocol stack. One of such solutions is Ethernet/IP [9], where IP stands for "Industrial Protocol".

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 commodity Ethernet technologies.

In this paper we propose an approach for assessing the timeliness characteristics of Ethernet/IP-based distributed systems. This approach builds upon modelling, simulation and statistical analysis of simulation results, and is part of a wider framework

This work was partially funded by Rockwell Automation under research contract INDEPTH and by FCT under CIDER and MethoDES Projects (POSI/1999/CHS/33139, POSI/2001/37334).

related to the research project INDEPTH − INDustrial-Ethernet ProTocols under Holistic analysis (www.hurray.isep.ipp.pt/indepth).

The timeliness analysis of a system is usually exploited in a framework dominated by the notion of absolute temporal guarantees. In those 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.

To this end, researchers usually follow two, generally alternative, approaches. These two approaches are based on:

1. simulation models of system components that mirror the actual behaviour of the system;
2. analytical models that give a measure of worst-case system latencies.

Each of those has advantages and disadvantages, when compared to each other. Simulation-based models can be applied to virtually all problems, and system details can be embodied into the models up to the desired level. However, a major drawback may turn out to be the time required in executing the simulation for large and realistic systems, particularly when results with high accuracy (narrow confidence intervals) are desired. Also, typically, simulations require the use of simulation development and deployment tools that entail difficulties or are not appropriate to be applied to the target system.

These drawbacks do not exist to the same extent in analytical-based approaches. However, and for complex distributed systems, analytical-based models tend to be overwhelmed with simplifications that often lead to very pessimistic assumptions, and therefore to very pessimistic worst-case results. Even knowing that a number of existing techniques may potentially be used and adapted to reduce this pessimism level, the benefit may appear at the cost of adding rather complex abstractions, such as precedence relationships [10], event phasing [11] and inheritance of time characteristics [12]. These, unfortunately, may lead to intractable mathematical models, thus making it further difficult to handle and reason the analytical abstractions.

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.

Therefore, in this paper we exploit the approach based on simulation models of system components that mirror the actual behaviour of the system.

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. Afterwards, we describe how we have been tackling the problem of modelling and simulating distributed systems based on that COTS technology. Finally, we discuss the use of simulation results to perform statistical timeliness analysis, by means of a concrete simulation example.

2. Ethernet/IP-based Distributed Systems

In CIP-based networks, such as Ethernet/IP, the majority of the messaging performed is done through connections. CIP connections define the packets that will be produced on the network, and can be of two types: Explicit Messaging or Implicit Messaging.

Implicit messaging is the messaging used for time critical I/O data, and therefore will receive the focus of our attention, specially the Cyclic Implicit CIP type of connections. A device produces cyclic messages on a predetermined rate basis, defined by the Requested Packet Interval (RPI) parameter. Underlying these transactions is a producer/distributor/consumer model, also usually found in other factory communication networks such as WorldFIP [13]. In Ethernet/IP networks the distribution is supported upon multicast UDP/IP that, in turn, is mapped onto Ethernet multicast.

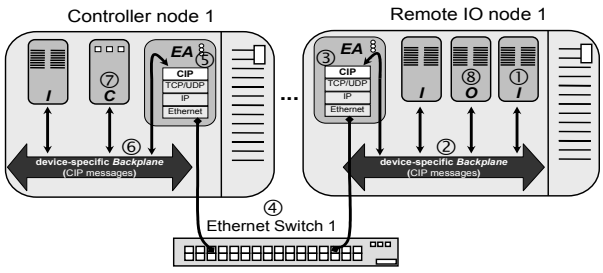

Figure 1: Ethernet/IP-like networks basic nodes and an end-to-end transaction example.

Ethernet/IP networks are constituted by three structuring types of nodes: Remote I/Os, Controllers and interconnecting Switches. These nodes communicate with each other via Ethernet. Diverse modules can compose the Remote I/O and Controller nodes. These modules communicate among them via a

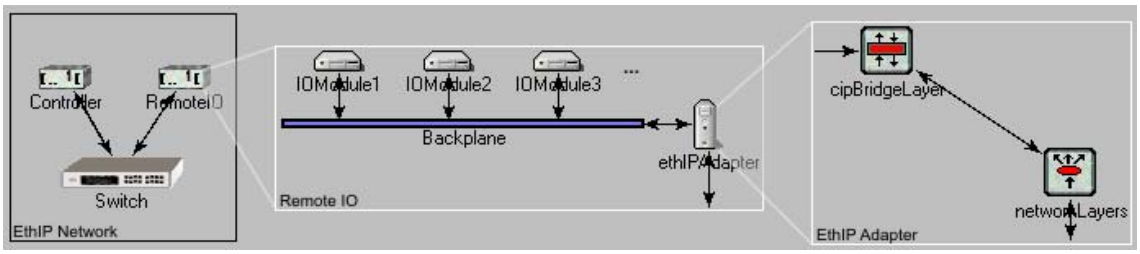

Figure 2: OMNET++ hierarchical models.

device-specific backplane (Figure 1). 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 node has no Controller modules.

Assuming the simple network scenario given in Figure 1, let us take a closer look to the type of end-to-end transactions we are addressing. A typical 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 then arrive at the EA, where it is processed and sent via the communication interface (③) to the Ethernet switch, that forwards the message to the corresponding output port(s) (④). The message will arrive to the Controller EA (⑤), where it is dispatched to the Controller module via the node backplane (⑥). At the controller (consumer of the data associated to the transaction), the input data will be processed by a task, that generates the related 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 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).

3. Ethernet/IP Simulation Model

The Ethernet/IP distributed system simulation environment was developed using the OMNeT++ [14] 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 characterized 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 (an OMNeT++ specific scripting language). An OMNeT++ model consists of hierarchically nested Modules (see Ethernet/IP example as depicted in Figure 2). These modules can have parameters which are used to customize 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 parameterization are considered into the design to a sufficient extent. In the next subsections, further details are provided concerning model implementation aspects.

3.1 The remote IO node

The Remote IO is composed of several IO modules and an Ethernet/IP Adapter, which communicate through a backplane, using CIP packets. 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 as both at the same time. 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 2).

The *Backplane* is a simulation module that exists both at Controller and Remote IO nodes. For simulation performance, at initialisation time the Backplane uses the information about the data connections produced/consumed at each module to build a table with information on the gates where to deliver each of the configured connections. Figure 3 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. 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 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 3: Backplane 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 (MAC) 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 receives the CIP messages from the Backplane and, in the CIP Bridge Layer (cipBridgeLayer in Figure 2) encapsulates them into UDP packets which are passed down to the Network Layer of the UDP/IP stack. On the opposite direction the packets are retrieved from 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 delays resulting from the concurrent access to the adapter resources. Figure 4 illustrates the NED definition of the Ethernet Adapter OMNeT++ module (a compound module). 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 4: EthIPAdapter NED definition.

The *connectionIDProducedList* and the *connectionIDConsumedList* parameters are used for listing the CIP connection identifiers of the connections produced and consumed in the node's modules connected to the backplane. The sub-modules of 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 4) are between these two layers and the input/output gates from the backplane and the Ethernet network.

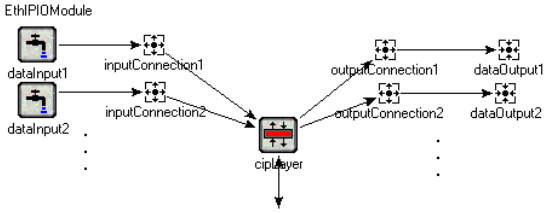

Figure 5: ONNET++ EthIPIOModule composition.

Each of the IO modules (labelled *IOModule1*, *IOModule2*, *IOModule3*, etc., in Figure 2) inside a node and connected to the Backplane contains a CIP Layer, responsible for managing data transfers 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 5).

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 parameterized hardware delay. This is illustrated in Figure 6, 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 6: IOConnection class message handler C++ code.

The data input generators (*dataInput1*, *dataInput2*, ..., in Figure 5) model the signals applied at the input pins of the IO. They are parameterized by the length of the data generated and the periodicity of the data generation, and by two delays introduced after the generation of the input (a hardware delay and a filter delay). OMNet++ supports defining any of these parameters as a user-defined randomly distributed function. 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 initialization file that may assign the parameters individually for each module in the simulation.

Figure 7 exemplifies the definition of the *dataInput* (NED code) parameters in an IO module: a random variable with a uniform distribution in the interval [100, 150] milliseconds.

Figure 8 illustrates the alternative setting of 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*).

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

Figure 7: OMNET++ EthIPIOModule NED code for parameter configuration.

```
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 8: EthIPIOModule parameter configuration through initialization 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. Of course, it is possible to parameterize each of the modules differently, and therefore manipulate their actual behaviour.

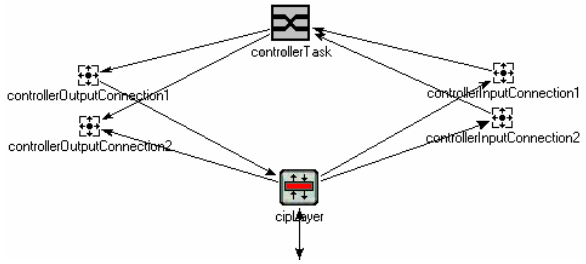

Figure 9: ONNET++ Controller module composition.

There is however a module that must be specified for the particular case of Controller nodes: the *Controller* module (Figure 9). In an actual Ethernet/IP system, the controller module is responsible for executing the tasks performing the control functions.

The Controller was modelled reusing some OMNeT++ modules described earlier: the IO

Connection modules and the CIP Layer. The *controllerInputConnection* module receives the data to be delivered to the *ControllerTask* module, corresponding to an output connection at the remote source node. The output data generated by the controller task is delivered to the *controllerOutputConnection* module. The ControllerTask (worst-case) response time is a parameter which is a time span introduced between the reception and the generation of the data. This parameter can be defined has a random function that best models the response time for each controller task.

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 recognizes 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 10. 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 10: Ethernet Switch 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 11 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 11: 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 [15], 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 initialization 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 parameterized 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 analyzed are then discussed in this section. Note that we are aiming at obtaining an estimation of the worst-case end-to-end response time for a number of transactions. A primary goal is to consider some fundamental aspects about the analysis of the simulation results.

4.1 Example scenario

The example system is constituted of three Remote IOs, one Controller and an interconnecting switch (Figure 12).

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 IOs to the Controller (Input direction) and the other half, from the Controller to the Remote IOs (Output direction).

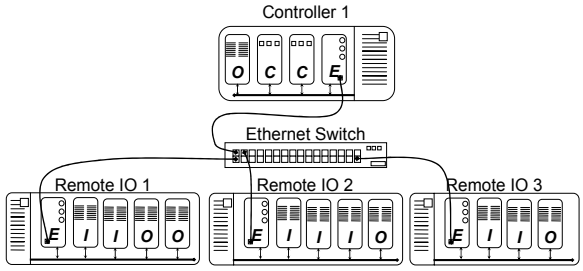

Figure 12: Example of simulated system.

Table 1 presents the identifiers of the system's connections whereas Tables 2-4 provide the details about the mapping of connections to the system modules.

Table 1. End-to-End transactions

Transaction	*Connection: Input Direction*	*Connection: 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

As an example, Transaction 8 is initiated at the IOModule3 of RemoteIO1 (connection 171) with an RPI of 200 ms (Table 2). It is delivered to module 2 of the Controller (Table 3), where the data is processed, and the corresponding output is generated (connection 181). This connection is then sent to the IOModule2 of RemoteIO3 (Table 4). The RPI of the output connections is equal to the corresponding input connection.

Table 2. Input Connections

Node	*Module*	*Input Connection*	
		ID	*RPI (ms)*
Remote IO 1	*IO module 3*	171	200
		172	350
Remote IO 2	*IO module 1*	131	10
	IO module 2	132	7
	IO module 3	133	25
		134	20
Remote IO 3	*IO module 1*	151	55
		152	80
		153	75

Table 3. Connections at the controller

Module	*Input Connection*	*Output Connection*
Controller 1 module 1	131	141
	132	142
	133	143
	153	163
Controller 1 module 2	134	144
	151	161
	152	162
	171	181
	172	182

Table 4. Output connections

Node	*Module*	*Output Connections*
Remote IO 1	*IO module 1*	141; 142; 14;
	IO module 2	163
Remote IO 2	*IO module 4*	144; 163
Remote IO 3	*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 instantiations 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 [16]. 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, such as in [17], 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 [16], rely on the fact that although the simulation results of a single simulation run are not independent, it is possible to obtain independent observations across the results of several 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.

The *replication/deletion* method is a fairly simple approach, with a reasonably good statistical performance [16], 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, is an approximately unbiased point estimation for *v*, and an approximate $100(1-\alpha)$ percent confidence interval for *v* may be obtained by [16]:

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

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

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

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

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

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 5 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.

The *X* in the table 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.

Table 5. 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 ,
Tr. 6	220.90 ± 6.27	[214.62 ,
Tr. 7	110.20 ±12.79	[97.41 ,
Tr. 8	400.74 ± 7.44	[393.30 ,
Tr. 9	700.59 ± 8.72	[691.87 ,

This evaluation of the behaviour of a concrete system may be of relevance to the systems designer, when a probabilistic analysis of the system is being carried out.

Note that this evaluation is more suitable for means and variance behaviour. Its applicability for values on the tail of distributions (such as worst-case) is still object of current work, thus the reader is referred to [18] for further discussion on these issues .

Some additional 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 more appropriate estimates to real time system parameters, analysis techniques that consider the particular statistical properties of these parameters must be applied.

6. References

[1] E. Tovar, F. Vasques, and A. Burns, "Communication Response Time in P-NET Networks: Worst-Case Analysis Considering the Actual Token Utilisation," Real Time Systems Journal, Kluwer Academic Publishers, vol. 22, pp. 229-249, 2002.

[2] L. M. Pinho and F. Vasques, "Reliable real-time communication in CAN networks," IEEE Transactions on Computers, vol. 52, pp. 1594-1607, 2003.

[3] T. Skeie, S. Johanssen, and O. Holmeide, "The Road to and End-to-End Deterministic Ethernet", in proceedings of the 4th IEEE International Workshop on Factory Communication Systems (WFCS'02), Vasteras, Sweden, pp. 3-9, 2002.

[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 Euromicro Conference on Real-Time Systems, Stockholm, Sweden, pp. 19-22, 2000.

[5] Rockwell Automation, "Making Sense of e Manufacturing: a Roadmap for Manufacturers", Rockwell Automation Inc., Cleveland, Ohio, White Paper 2000. Available online at www.rockwellautomation.com.

[6] Y. Song, A. Koubaa, and F. Simonot, "Switched Ethernet for Real-Time Industrial Communication: Modelling and Message Buffering Delay Evaluation", in proceedings of the 4th IEEE International Workshop on Factory Communication Systems (WFCS'04), Vasteras, Sweden, pp. 27-35, 2002.

[7] J. Georges, E. Rondeau, and T. Divoux, "Evaluation of switched Ethernet in an industrial context by using the Network Calculus", in proceedings of the 4th IEEE International Workshop on Factory Communication Systems (WFCS'02), Vasteras, Sweden, pp. 19-26, 2002.

[8] M. Volz, "Quo Vadis Layer 7", The Industrial Ethernet Book, vol. 5, 2001, 8-10. Available online at http://ethernet.industrial-networking.com/.

[9] Paul Brooks, "Ethernet/IP - Industrial Protocol", in proceedings of the 8th IEEE International conference on Emerging Technologies and Factory Automation, Antibes - Juan les Pins, France, pp. 505-514, Volume 2, 2001.

[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 The 20th IEEE Real-Time Systems Symposium (RTSS'99), Phoenix, Arizona, pp. 328-339, 1999.
[11] J. C. Palencia Gutierrez and Michael Gonzalez Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets", in proceedings of the The 19th IEEE Real-Time Systems Symposium (RTSS'98), Madrid, Spain, pp. 26-37, 1998.
[12] Ken Tindell, "Holistic schedulability analysis for distributed hard real-time systems," Microprocessors and Microprogramming, Elsevier Science Publishers, vol. 50, pp. 117-134, 1994.
[13] L. Almeida, E. Tovar, J. Fonseca, and F. Vasques, "Schedulability Analysis of Real-Time Traffic in WorldFIP Networks: an Integrated Approach," IEEE Transactions on Industrial Electronics, vol. 49, pp. 1165-1174, 2002.
[14] A. Varga, "OMNeT++ Discrete Event Simulation System", v2.3, 2004. Web Site: http://www.omnetpp.org/.
[15] S. Deering, "Host Extensions for IP Multicasting", RFC 1112: Stanford University, 1989.
[16] Averill M. Law and W. David Kelton, Simulation modeling and analysis, 3rd ed. New York: McGraw-Hill, 2000.
[17] A. Wall, J. Anderson, 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 (ISOORC'03), Hokkaido, Japan, pp. 257-268, 2003.
[18] N. Pereira, E. Tovar, B. Baptista, L. M. Pinho, and I. Broster, "A Few What-Ifs on Using Statistical Analysis of Stochastic Simulation Runs to Extract Timeliness Properties", Polytechnic Institute of Porto, Porto, Portugal, Technical Report, July 2004. Available online at http://www.hurray.isep.ipp.pt/indepth.