Performance and Reliability of Fault-Tolerant Ethernet Networked Control Systems

Ramez M. Daoud[1], Hassanein H. Amer[1]
and Hany M. ElSayed[2]
[1]*American University in Cairo*
[2]*Cairo University*
Egypt

1. Introduction

In many control applications, networks are being used as a transmission medium for control data such as sensor readings, controller signals, and alarm signals. The resulting control system is termed a Networked Control system (NCS) (Clauset et al., 2008; Hespanha et al., 2007; Yang, 2006). Examples of NCS application areas include industrial automation, building automation, home automation, intelligent vehicle systems, and advanced aircraft and spacecraft. Compared to point-to-point wiring, this approach simplifies wiring in complex systems where several subsystems are interconnected and where sensors and actuators may be physically remote from the controller. System hence becomes easier to control and maintain. Networks also enable communication between several control loops and fault-tolerance through redundancy of components. This chapter summarizes work done by the authors in the area of performance and reliability of networked control systems.
Communication networks were first introduced in digital control systems in the 1970's. Since then, several types of communication networks have been developed to serve this field. Protocols for these networks can be grouped into *fieldbuses* (e.g. FIP and PROFIBUS), *automotive buses* (e.g. CAN), other *machine buses* (e.g. 1553B and the IEC train communication network), *general-purpose networks* (e.g. IEEE LAN's and ATM-LAN) and a number of *research protocols* (e.g. TTP).
In manufacturing applications, the network connecting controllers with sensors and actuators typically constitutes one level in a hierarchy of networks. Fig. 1 illustrates a general network hierarchy model (Lian et al., 2001b). This model consists of five levels, each one having different goals and also different communication capabilities, protocols and complexity. Level one is the device or sensor-actuator level that is used to interconnect controllers, sensors or actuators. Level two is the cell control level and it is designed to be used with cell controllers such as at milling, lathe and control workstations in manufacturing plants. Generally, levels one and two are called *sensor* and *fieldbus*, respectively. Level three is the supervisory level and is used to interconnect machine cells that perform different manufacturing processes. Level four is the plant management level and is used to coordinate various tasks executed inside a plant such as manufacturing

engineering, production management, and resource allocation. Level five is the corporate management level. It may interconnect workstations located in different cities or countries (Lian et al., 2001b).

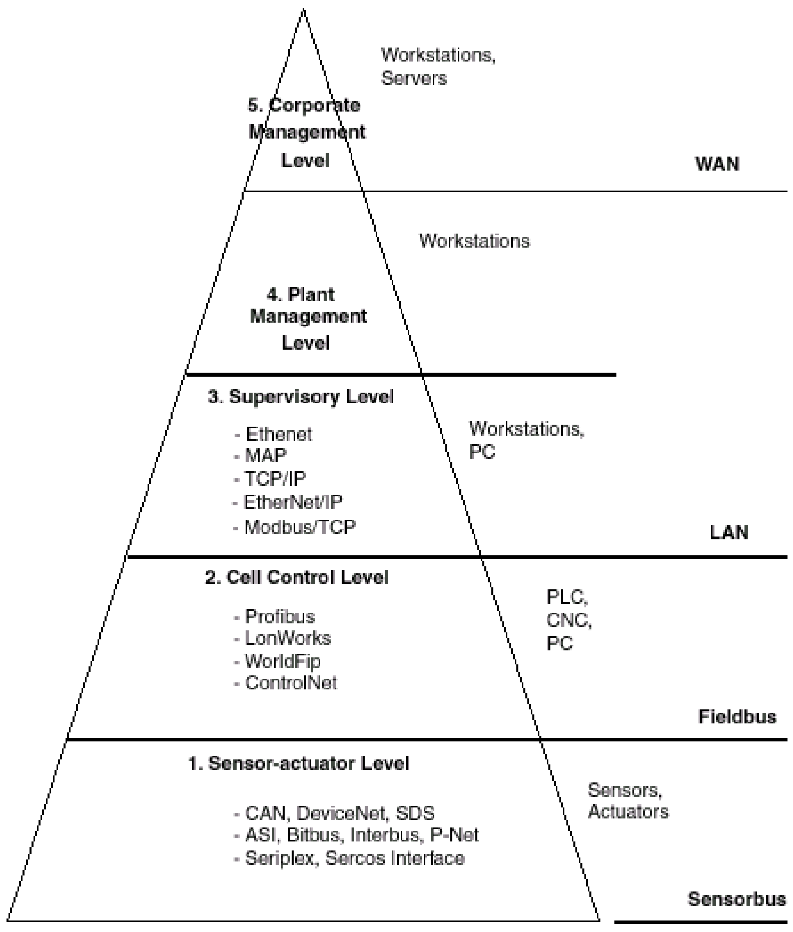


Fig. 1. A general network hierarchy model (Lian et al., 2001b)

A wide variety of network protocols can be used to build an NCS, each suitable for a particular application sector (Thomesse, 2005). However, the use of Ethernet remains a viable and interesting option (Decotignie, 2005). Ethernet is now the dominant local area networking solution in the home and office environments. It is fast, economic and easy to install. Many computerized equipments now come with built-in Ethernet Interfaces. These are some of the reasons why a number of manufacturers of industrial control systems are now migrating to the use of Ethernet on the production floor and integrate it with the management floor (Decotignie, 2005).

A highly desirable requirement is for the office Ethernet communication capability to be fully retained when applying it for control, i.e., the best solution would be if no protocol change were introduced (Daoud et al., 2003; Felser, 2005). Capabilities of Ethernet networks allow us to envision the merger of several hierarchy levels in a single network. A main focus of this research is thus to test the network operation and performance in the presence of mixed traffic.

A major problem in networked control systems is the delays introduced by the network in the control loop. Further problems may be caused by possible loss of data packets. Beyond

certain limits, delays will result in poor system response and tend to destabilize the control loop. Network delays are generally variable and depend on such factors as the used network protocol, network topology, and the amount of network traffic from other sources.
In this chapter the focus is on both the performance and reliability aspects of Ethernet based NCS. The rest of the chapter is organized as follows: Section 2 presents a small survey on previous works done in the area of NCS using Ethernet as communication protocol. Section 3 Introduces the network and its different components. Section 4 describes the model for simulation. Section 5 presents the network simulation results. Sections 6, 7 and 8 discuss the reliability and availability of fault-tolerant production lines. Section 9 concludes this research.

2. Ethernet Networked Control Systems

Abundant research results on the use of Ethernet as a communication network for NCS have been published in the recent years. Interested readers may refer to (Brahimi, 2007; Decotignie, 2005; Felser, 2005; Georges, 2005; Kumar, 2001; Lian et al., 2001a; Marsal, 2006a; Nilsson, 1998; Skeie et al., 2002).

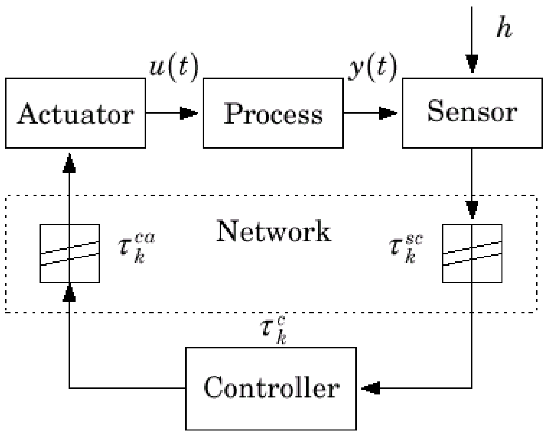


Fig. 2. NCS Block Diagram (Nilsson, 1998)

Automated workcells (Morris, 2005) always include sensors, controllers and actuators connected over the network. The control packet flow on the Ethernet channel is based on publisher/subscriber mode of communication: once the packet is generated, it can be heard by any node on the network. This facilitates the data flow and eliminates the duplication of sensors. A comparison between different methods of communication is given in (Marsal, 2006a). The model of Fig. 2 (Nilsson, 1998) shows a schematic of NCS.
In this model, the physical data is sensed on a periodical basis (clock driven) every *h* seconds. It is transmitted from the sensor to the controller of the network facing a delay τ_k^{sc}.
It is consumed at the controller node and processed over τ_k^{c} delay. The controller sends the control action over the network to reach the actuator after a delay τ_k^{ca}.
The source of non-determinism in switched Ethernet is queuing delays. For example, the controller node may generate non-real-time traffic (also known as explicit messaging, in contrast with implicit messaging used to designate the real-time control) like FTP sessions or HTTP. This will perturb the queues and the processing loads at the controller node.

Accordingly, the end-to-end delays (the delay measured from the sensor node to the actuator node taking into consideration all kind of encapsulation/de-capsulation, processing and propagation delays) will not be constant. This is what is called mixed traffic environment. It is important to test the Ethernet NCS behavior in simple control environment (only control packets are communicated) and mixed traffic environment.

Early works such as (Meditch & Lea, 1983) tried to modify the medium access sub-layer of CSMA/CD to distinguish between real-time and other traffic packets. Studies were conducted to test stability of the communication channel and to optimize its performance.

Rockwell-Automation studied the use of Ethernet in its switched topology in control and they merged Ethernet with ControlNet to make what is called EtherNet/IP (Ethernet/IP; Lounsbury & Westerman, 2001; ODVA1; ODVA2; ControlNet). By using both TCP/IP and UDP/IP protocols to encapsulate networked messages, both real-time I/O and "explicit messaging" can occur. Also, by providing Ethernet users with real-time I/O, device-configuration, and diagnostic capabilities, along with interoperability and interchangeability, EtherNet/IP provides an Ethernet standard for automation (Ethernet/IP).

In (Walsh & Ye, 2001), a new dynamic scheduling technique for NCS is proposed. The network here is not only dedicated for control purposes, but it can also accommodate communication frames. This gives rise to network induced delays due to unpredictable loads. The control algorithm was made off-line ignoring network delays. This simplified the analysis tremendously. Including time delays is a new approach to validate their work. Also, the simplicity of this approach makes it attractive to be used in general studies for any NCS. The simplicity of the approach in (Walsh & Ye, 2001) comes from the fact that they are using a simple state space representation of the overall NCS.

In (Marsal, 2006a), an analysis is made to define the source of delay in an Ethernet NCS; it shows that the overall response time is the sum of three delays: processing time, waiting time for synchronization of asynchronous processes and waiting time for availability of shared resources. A comparison between two methods to evaluate this response time (simulation and colored Petri nets) can be found in (Marsal et al., 2006b). In this research, the synchronization delays are included as well as the time for availability of shared resources in the processing delays of the nodes. Also, these two major time delays are the focus of other research works (Sundararaman et al., 2005).

In this chapter, the focus is only on switched Ethernet at different speeds. In (Skeie et al., 2002), Fast Ethernet was tested to eliminate the various, usually incompatible, communication networks at the traditional substation automation. This study was conducted to test the possibility of using Fast Ethernet in the switched topology in power station control application. Results of this study were satisfactory within the time frame of the considered application. Because the application presented in (Skeie et al., 2002) had relatively large time frame limit, Fast Ethernet switch topology succeeded to run this system. Later works showed that with more tight timing requirements, especially in mixed traffic environment, the speed of Gigabit Ethernet will be necessary for successful operation.

3. Network Nodes

The network nodes of a typical NCS model are namely: sensors, actuators and controllers. These are the active nodes that generate and consume traffic. Other nodes that are present to build the network are switches for a switched operation mode. The fabric of the network used in this research is mainly Ethernet at 100Mbps and 1Gbps speeds. This means that there are two types of networks that are tested in this study: Switched Fast Ethernet (100Mbps Ethernet) and Switched Gigabit Ethernet (1Gbps Ethernet).

3.1 Sensor/actuator Networking Level

At this level, networked devices consist essentially of smart sensors, networked controllers, and smart actuators. Smart sensors are nodes that have the capability of data acquisition, intelligence and communication. They acquire proper physical data such as temperature or speed from the industrial environment and have a network-capable application processor to interface with the network. Intelligence gives smart sensors the ability to function independently. Finally to be able to communicate over the network, the sensor must be able to properly encode the information before sending it out on the network.

Smart actuators have the features of actuation, intelligence and communication. They are able to decode the information from the network medium and apply it to the physical devices (Lian et al., 2001b).

Networked controllers have the major function of analyzing the sensor data, making decisions, and giving commands to actuators. The control algorithm should handle decentralized information analysis as well as traditional centralized analysis. Networked controller nodes may also provide a human-machine interface to operators or higher-level managers.

Candidate networks protocols at this level must meet two main criteria: bounded time delay and guaranteed transmission. Unsuccessfully transmitted or large time-delay messages may deteriorate system performance. The system can even become unstable. Several protocols have been proposed to meet these requirements for control systems (Nilsson, 1998). The performance requirements mentioned above are used to determine the capability of the network medium and to provide design specifications to control parameters such as sampling rates as well as network parameters such as communication rates.

3.2 Using Ethernet in the Sensor/Actuator Level

With the use of Ethernet at this level, many things that were not possible in past implementations of NCS will be enabled. Once the industrial floor (the machines network connection) is running on top of Ethernet, it can be interconnected with the management floor (engineering and management network connections). This will help in problem diagnostic and set-up. Now more and more functions can be added. One possibility is on-line system diagnostics and fix-up, by logging into the machine while running in normal operation and setting-up some parameters without the need to stop the operation. This means integration of communication packets (log-on, request/download file, up-load file, log-off) while performing the usual control tasks (traffic of real-time control packets). Moreover, some tasks that can be performed by the operator can be enabled like web-browsing and e-mail check. These tasks add to the communication load that the network handles as an overhead to the pure control load that it is built to support.

3.3 Performance Metric

The performance metrics of network systems that impact control system requirements include: access delay, transmission time, response time, message delay, message collisions (percentage of collision), message throughput (percentage of packets discarded), network utilization, and determinism boundaries. For control systems, candidate control networks generally must meet two main criteria: bounded time delay and guaranteed transmission; i.e., a message should be transmitted successfully within a bounded time delay (Lian et al., 2001b). Unsuccessfully transmitted or large time-delay messages from a sensor to an actuator may deteriorate system performance or make a system unstable. Several protocols have been proposed to meet these requirements for control systems (Nilsson, 1998). The performance metrics mentioned above are used to determine the capability of the network medium and to provide design specifications to control parameters such as sampling rates as well as network parameters such as communication rates (Daoud et al., 2004a).

As in (Georges, 2005), the focus of this research is to use Ethernet IEEE802.3 Std without modifications. A previous study was made to use Ethernet in control by changing the frame structure for real time packets (Tolly, 1997). Another study was made to design a real-time controller to control traffic of the communication medium in case of real-time constraint (Lee & Cho, 2001). More research can be found in (Brahimi et al., 2006; Eker & Cervin, 1999; Georges et al., 2006; Jasperneite & Elsayed, 2004a; Lian et al., 2001a; Vatanski et al., 2006; Wang & Keshav, 1999; Wittenmark et al., 1998; Zhang et al., 2001).

In this research, the system success or failure is evaluated based on measuring the delay faced by the sensor data traveling over the network to reach the controller, the processing delay at the controller node, the propagation delay from the controller to the actuator node faced by the control packet, and finally the processing delay at the actuator node before applying the control word to the physical process. This end-to-end delay takes into consideration all kind of data encapsulation, propagation, de-capsulation and processing in all nodes on the network. End-to-end delay for Ethernet NCS can also be analyzed with network calculus as in (Georges, 2005; Grieu, 2004). The network delay can be expressed as:

$$D_T = D_T\Big|_{\text{controller}} + D_T\Big|_{\text{actuator}} \tag{1}$$

where D_T is the total end-to-end delay.

$$D_T\Big|_{\text{controler}} = T_{ps}\Big|_{\text{sensor}} + T_{encap}\Big|_{\text{sensor}} + T_p\Big|_{\text{sensor}\rightarrow\text{controller}} + T_q\Big|_{\text{controller}} + T_{decap}\Big|_{\text{controller}} \tag{2}$$

$$D_T\Big|_{\text{actuator}} = T_{ps}\Big|_{\text{controller}} + T_{encap}\Big|_{\text{controller}} + T_p\Big|_{\text{controller}\rightarrow\text{actuator}} + T_q\Big|_{\text{actuator}} + T_{decap}\Big|_{\text{actuator}} \tag{3}$$

Where T_{ps} is the processing delay

T_{encap} is the encapsulation delay

T_p is the propagation delay

T_q is the queuing delay

T_{decap} is the de-capsulation delay

4. Models Description

Many control applications naturally run at very low speeds compared to the speeds of the new network standards. This may imply that the required delays of control packets can be met under realistic loading conditions without handling these packets in a special manner.
Individual machines can be transformed into automated workcells (Soloman, 1994). Networked Control Systems (NCS) make it possible to transform machines into small networks (machine LANs) (Daoud et al., 2003). All sensors are sources of traffic. All actuators are sinks of traffic. Data produced at sensor nodes is sent over the machine network to reach the controller node. At the controller, control information is computed and sent once again over the network to reach the corresponding actuator node. At the actuator, control action is applied on the physical system (the plant).
Sensors and actuators in this scheme are smart. Smart sensors as well as smart actuators have the capability of data encapsulation/de-capsulation. They have network capability to be able to communicate over the machine network. Smart sensors are clock driven nodes: data is sampled at the sensor nodes at constant frequency. This frequency can vary from one sensor to the other depending on the physical quantity it is sensing (temperature is sampled at a lower sampling frequency than speed for example). Once the data is ready at the sensor node, it is encapsulated in network packet format and sent over the machine network.
The controller node, which is an Industrial Personal Computer (IPC), is event driven. When it receives a packet form a sensor and after its de-capsulation and error check, it starts computing the necessary control action. Again, it encapsulates the control word and sends it in packet format to the actuator node. A smart actuator receives the control word and applies the appropriate control action to the system after de-capsulation and error check. Smart actuators are event driven as well (Daoud & Amer, 2007).
Two main Models are presented in this work. The Stand Alone Machine Model and the In-Line Production Model.

4.1 Stand Alone Machine Model Description

Stand Alone Machine Model tests the operation of a single machine (workcell) having all its connections based on Ethernet to implement the Ethernet NCS model.

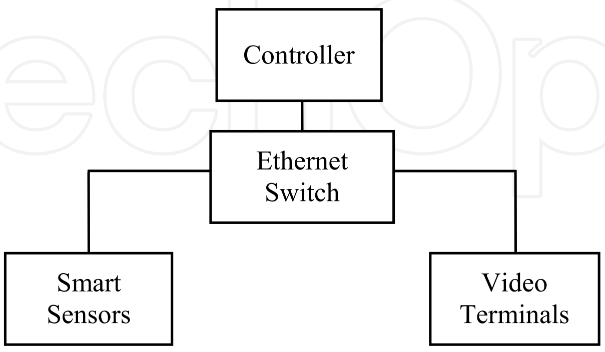


Fig. 3. Stand alone machine model

Two models were built for this study. One model is run on-top-of Fast Ethernet and the other one is run over Gigabit Ethernet for performance comparison. One model consists of 16 sensors, one controller, and 4 actuators, based on the model of (Skeie et al., 2002). In the following, this model will be referred to as the *light traffic system*. The other model consists of 48 sensors, one controller, and 4 actuators. This model will be referred to as the *heavy traffic system*.

Sensors and actuators are smart. For traditional control using PLCs, 1 revolution per second is encoded into 1,440 electric pulses for electrical synchronization and control. This is why, the system presented in this study is operating at a sampling frequency of 1,440 Hz. Consequently, the system will have a deadline of 694 μs, i.e., a control action must be taken within a frame of 694 μs as round-trip delay originating from the sensor, passing through the controller, and transmitted once more over the network to reach the actuator.

It should be noted that the heavy traffic case should be accompanied by an increase in the processing capabilities of the controller itself. Thus while in the light traffic case the controller was able to process 28,800 packets per second, this number was increased to 74,880 in the heavy traffic case. (These numbers result from multiplying the number of sources and sinks by the sampling rate). The packet delay attributable to the controller will thus be reduced in the heavy traffic case.

OPNET (Opnet) was used as a simulation platform. Real-time generating nodes (smart sensors and smart actuators) were modeled using the "advanced workstation" built-in OPNET model. This model allows the simulation of a node with complete adjustable parameters for operation. The node parameters were properly adjusted to meet the needed task as source of traffic (smart sensor) or sink of traffic (smart actuator). The Controller node was simulated also using "advanced workstation". The Controller node is the administrator in this case: it receives all information from all smart sensors, calculate control parameters, and forward control words to dedicated smart actuators. Producer/ Customer model is finally used to send data from Controller node to smart actuators.

All packets were treated in the switch in a similar manner, i.e., without prioritization. Thus, the packet format of the IEEE 803.2z standard (IEEE, 2000) was used without modification.

Control signals in the simulations are assumed to be UDP packets. Also, the packet size was fixed to minimum frame size in Gigabit Ethernet (520 bytes).

Simulations considered the effect of mixing the control traffic with other types of traffic. These include the option of on-line system diagnostic and fix-up (log-on, request/ download file, up-load file, log-off) as well as e-mail and web-browsing. FTP of 101KB files was considered (Skeie et al., 2002). HTTP, E-mail and telnet traffic was added using OPNET built-in heavy-load models (Daoud et al, 2003).

4.2 In-Line Production Model Description

In many cases, a final product is not produced only on one machine, but, it is handled by several machines in series or in-line. For this purpose, the In-Line Production Model is introduced and investigated. The idea is simply connecting all machine controllers together. Since each individual machine is Ethernet based, interconnecting their controllers (via Ethernet) will enable them to have access to the sensor/actuator level packet flow.

The main function of the controller mounted on the machine is to take charge of machine control. An added task now is to help in synchronization. The controller has the major role of synchronizing several machines in line. This can also be done by connecting the networks of the two machines together. To perform synchronization, the controller of a machine sends its status vector to the controller another machine, and vice versa. Status vector means a complete knowledge of machine information, considering the cam position for example, the

production rate, and so on. These pieces of information are very important for synchronization, especially the production rate. This is because, depending on this statistic, the machines can speed up or slow down to match their respective productions.
A very important metric also, is the fact that the two controllers can back-up data on each other. This is a new added feature. This feature can achieve fault tolerance: in case of a controller failure, the other controller can take over and the machine is not out of service. Although this can slow down the production process, the production is not stopped (Daoud et al., 2004b). Hardware or software failure can cause the failure of one of the controllers. In that case, the information sent by the sensors to the OFF controller is consumed by another operating controller on another machine on the same network (Daoud et al., 2005). "OFF" controller is used instead of failed because the controller can be out of service for preventive maintenance for example. In other words, not only failure of a controller can be tolerated, but regular and preventive maintenance also; because in either cases, failure or maintenance, the controller is out of order.

5. OPNET Network Simulations & Results

First, network simulations have to be performed to validate the concept of Ethernet integration in its switched mode as a communication medium for NCS. OPNET is used to calculate system performance.

5.1 Stand Alone Machine Models Simulation Results

For the light traffic system, and integrating communication as well as control traffic, results for Fast Ethernet are found to be 671 μs round-trip delay in normal operating conditions, and 683 μs round-trip delay as peak value. Results for Gigabit Ethernet are found to be 501 μs round-trip delay in normal operating conditions, and 517 μs round-trip delay as peak value. As the end-to-end delay limit is set to 694 μs (one sampling period), it can be seen that 100Mbps Ethernet is just satisfying the delay requirements while 1Gbps Ethernet is excellent for such system (Daoud et al., 2003).
For the heavy traffic system that consists of 48 smart sensors, 4 smart actuators and one controller, results for Fast Ethernet are found to be 622 μs round-trip delay in normal operating conditions, and 770 μs round-trip delay as peak value. Results for Gigabit Ethernet are found to be 450 μs round-trip delay in normal operating conditions, and 472 μs round-trip delay as peak value. The round-trip delay limit is still 694 μs (one sampling period). It can be seen that 100Mbps Ethernet exceeds the time limit while 1Gbps Ethernet is runs smoothly and can accommodate even more traffic (Daoud et al., 2003).
All measured end-to-end delays include processing, propagation, queuing, encapsulation and de-capsulation delays according to equation 2 (Daoud, 2008).

5.2 In-Line Production Light Traffic Models Simulation Results

The first two simulations consist of two light-traffic machines working in-line with one machine having a failed controller. The failed controller traffic is switched to the operating controller node. One simulation uses Fast Ethernet while the other uses Gigabit Ethernet as communication medium.

Other simulations investigate Gigabit Ethernet performance with more failed controllers on more machines in-line with only one functioning machine controller. In this case, the traffic of the failed controllers is deviated to the operational controller. Other simulations are run to test machine speed increase. As explained in the previous section, the nominal machine speed tested is 1 revolution per second (1,440Hz).
Non-real-time traffic (as in (Daoud et al., 2003)) is added in the three simulations. This is to verify whether or not the system can still function and also if it can accommodate real and non-real-time traffic.
Let the sensors/actuators of the machine with the operational controller be called **near** sensors/actuators. Also, let the sensors/actuators of the machine with the failed controller be called **far** sensors/actuators (Daoud, 2004a).
Results for Fast Ethernet indicate that the delay is too high. The real-time delay a packet faces traveling from the near sensor to the controller and then to the near actuator is around 732 μsec. This is the sum of the delay the real-time packet faces traveling from sensor to controller and the delay it faces traveling from controller to actuator. For the far sensors and actuators, the delay is again too large: around 827 μsec.
Results for Gigabit Ethernet indicate that the delay is small: Only 521 μsec round-trip delay for near nodes (see Fig. 4) and 538 μsec round-trip delay for far nodes.
For three machines with only one controller node operational and running on-top-of Gigabit Ethernet, a round-trip delay of approximately 567 μsec was found for near nodes and approximately 578 μsec round-trip delay for far nodes (Daoud et al., 2004b).
When non-real-time traffic (of the same nature discussed in (Daoud et al., 2003)) is applied in order to jam the control traffic in all three scenarios, a considerable delay is measured. This delay is too large and causes a complete system failure because of the violation of the time constraint of one sampling period. Because of the 3 msec delay that appears in these circumstances with 2 OFF controllers and only 1 ON controller, explicit messaging must be prevented. Explicit messaging here refers to a mixture of non-real-time load of HTTP, FTP, e-mail check and telnet sessions. This is in contrast with "implicit messaging" of real-time control load.

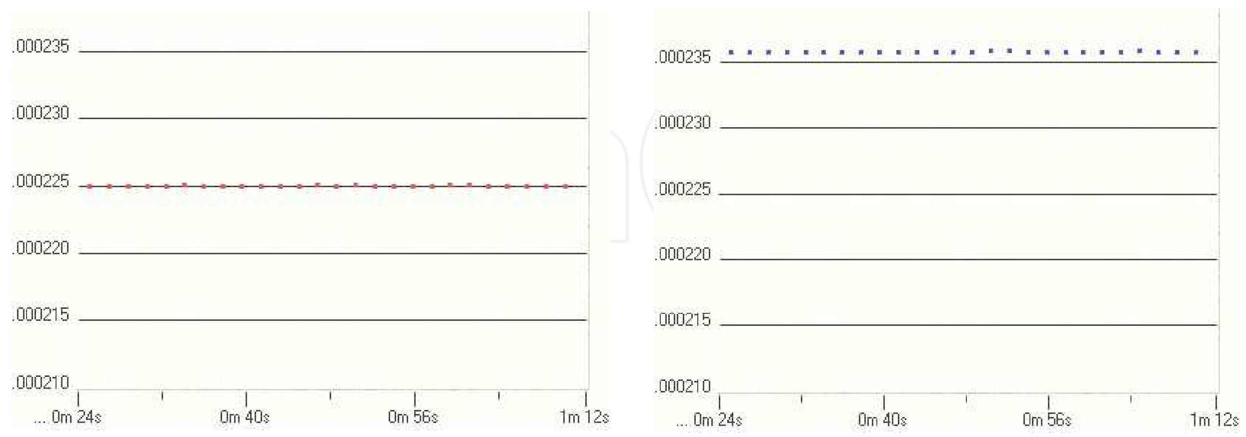
Machine Speed (rps)	Maximum Permissible Delay (μs)	Number of Machines	Number of OFF Controllers	Maximum Measured Delay (μs)
1	694	1	0	501
1	694	2	1	538
1	694	3	2	578
1	694	4	3	682
1	694	5	4	0.266s
1.2	579	3	2	536
1.2	579	4	3	545
1.3	534	2	1	509
1.3	534	3	2	534
1.3	534	4	3	545
1.4	496	1	0	476
1.4	496	2	1	501
1.5	463	1	0	476

Table 1. OPNET Simulation Results for In-Line Light Traffic Machine Model (Daoud et al., 2005)

This combination of non-real-time traffic loads simulates a real overhead jamming load introduced by the operator or chief engineer (specially FTP loads). This constraint is quiet acceptable in critical operation and preventing all kinds of non-real-time traffic is a justifiable sacrifice (Daoud et al., 2005). Final results are tabulated in Table 1.

5.3 In-Line Production Heavy Traffic Models Simulation Results

In this section, a simulation study of heavy traffic machines model consisting of 48 sensors, 1 controller and 4 actuators working in-line, is conducted using OPNET. This NCS machine is simulated as switched Star Gigabit Ethernet LAN. Sensors are sources of traffic. The Controller is an intermediate intelligent node. Actuators are sinks of traffic. Having 52 real-time packet generation and consumption nodes (48 sensors and 4 actuators) produces a traffic of 74,800 packet per second on the ether channel. This is because the system is running at a speed of 1 revolution per second (rps) to produce 60 strokes per minute (Bossar). Each revolution is encrypted into 1,440 electric pulses, which means that the sampling frequency is 1,440Hz (sampling period of 694μs). The number of packets (74,800) is the multiplication of the number of nodes (52) by the sampling frequency (1,440) (Daoud et al., 2003).
The most critical scenarios are studied. In these simulations, there is only one active controller while all other controllers on the same line are out of service. Studies for 2, 3 and 4 in-line production machines are done. In all simulations, only one controller is functional and accommodates the control traffic of all 2, 3, or 4 machines on the production line. It was found that the system can tolerate the failure of a maximum of 2 failed controllers in a 3-machine production line. In the case of a 4-machine production line with only one functional controller and 3 failed controllers, the deadline of 694μs (1 sampling period) is violated (Daoud & Amer, 2007).
Accordingly, it is again recommended to disable non-real-time loads during critical mode operation. In other control schemes that do not have the capabilities mentioned in this study, the production line is switched OFF as soon as one controller fails.



Delay at actuator node Delay at controller node

Fig. 4. OPNET Results for Two-Machine Production Line (Heavy Traffic)

In all cases, end-to-end delays are measured. These delays includes all types of data encapsulation/de-capsulation on different network layers at all nodes. They also include propagation delays on the communication network and the computational delay at the controller node. Results are tabulated in Table 2. Sample OPNET results are shown in Fig. 4.

Machine Speed (rps)	Maximum Permissible Delay (μs)	Number of Machines	Number of OFF Controllers	Maximum Measured Delay (μs)
1	694	2	1	461
1	694	3	2	522
1	694	4	3	1ms
1.1	631	2	1	497
1.1	631	3	2	551
1.2	579	2	1	464
1.2	579	3	2	473
1.3	534	2	1	483
1.3	534	3	2	520
1.4	496	2	1	476
1.4	496	3	2	553
1.5	463	2	1	464

Table 2. OPNET Simulation Results for In-Line Heavy Traffic Machine Model (Daoud & Amer, 2007)

6. Production Line Reliability

In the previous sections, fault-tolerant production lines were described and studied from a communications/control point of view. It was shown, using OPNET simulations, that a production line with several machines working in-line, can work in a degraded mode. Upon the failure of a controller on one of the machines, the tasks of the failed controller are executed by another controller on another machine. This reduces the production line's down time. This section shows how to estimate the Mean Time To Failure (MTTF) and how to use it to find the most cost-effective way of increasing production line reliability.

Consider the following production line; it consists of two machines working in-line. Each machine has a controller, smart sensors and smart actuators. The sampling frequency of each machine is 1,440 Hz.. The machine will fail if the information delay from sensor to controller to actuator exceeds 694 μsec. Also, if one of the two machines fails, the entire production line fails.

In (Daoud et al., 2004b), fault-tolerance was introduced on a system consisting of two such machines. Both machines were linked through Gigabit Ethernet. The Gigabit Ethernet network connected all sensors, actuators and both controllers. It was shown that the failure of one controller on either of the two machines could be tolerated. Special software detected the failure of the controller and transferred its tasks to the remaining functional controller. Non-real-time traffic of FTP, HTTP, telnet and e-mail was not permitted. Mathematical tools are needed to justify this extra cost and prove that production line reliability will increase. One such tool is Markov chains. This will be explained next.

6.1 Markov Model and Mean Time To Failure

Continuous-time Markov models have been widely used to predict the reliability and/or availability of fault-tolerant systems (Billinton & Allan, 1983; Blanke et al., 2006; Johnson, 1989, Siewiorek & Swarz, 1998; Trivedi, 2002). The Markov model describing the system being studied, is shown in Fig. 5. This same model is also found in (Arnold, 1973; Trivedi, 2002). State START is the starting state and represents the error-free situation. If one of the two controllers fails, the system moves from state START to state ONE-FAIL. In this state, both machines are still operating but only one controller is communicating with all sensors and actuators on both machines. If this controller fails before the first one is repaired, the system moves from state ONE-FAIL to state LINE-FAIL. This state is the failure state. The transition rates for the Markov chain in Fig. 5 are explained next.

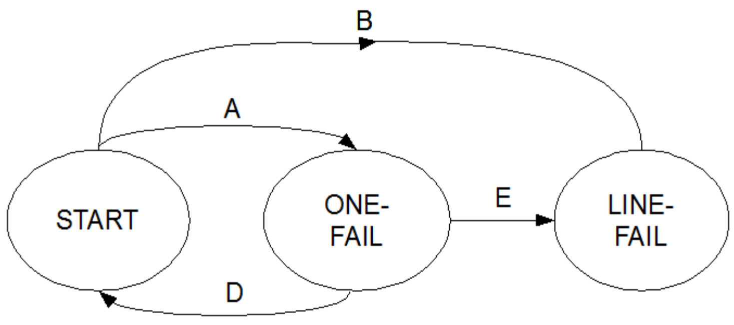


Fig. 5. Markov model

The system will move from state START to state ONE-FAIL when one of the two controllers fails, assuming that the controller failure is detected and that the recovery software successfully transfers control of both machines to the remaining operational controller. Otherwise, the system moves directly from state START to state LINE-FAIL. This explains the transition from state START to state LINE-FAIL. Let ***c*** be the probability of successful detection and recovery. In the literature, the parameter ***c*** is known as the *coverage* and has to be taken into account in the Markov model. One of the earliest papers that defined the coverage is (Arnold, 1973). It defined the coverage as the proportion of faults from which a system automatically recovers. In (Trivedi, 2002), it was shown that a small change in the value of the coverage parameter had a big effect on system Mean Time To Failure (MTTF). The importance of the coverage was further emphasized in (Amer & McCluskey, 1986, 1987a, 1987b, 1987c). Here, the controller software is responsible for detecting a controller failure and switching the control of that machine to the operational controller on the other machine. Consequently, the value of the coverage depends on the quality of the switching software on each controller.

Assuming, for simplicity, that both controllers have the same failure rate λ, the transition rate from state START to state ONE-FAIL will be equal to $A=2c\lambda$.

As mentioned above, the system will move from state START to state ONE-FAIL if a controller failure is not detected or if the recovery software does not transfer control to the operational controller. A software problem in one of the controllers, for example, can cause sensor data to be incorrectly processed and the packet sent to the actuator will have incorrect data but correct CRC. The actuator verifies the CRC, processes the data and the system fails. Another potential problem that cannot be remedied by the fault-tolerant architecture described here is as follows: Both controllers are operational but their inter-

communication fails. Each controller assumes that the other has failed and takes control of the entire production line. This conflict causes a production line failure. Consequently, the transition rate from state START to state LINE-FAIL will be equal to B=(1-c)2λ.
If the failed controller is repaired while the system is in state ONE-FAIL, a transition occurs to state START. Let the rate of this transition be D=μ. While in state ONE-FAIL, the failure of the remaining controller (before the first one is repaired) will take the system to state LINE-FAIL. Hence, the transition rate from state ONE-FAIL to state LINE-FAIL is equal to E=λ. The Markov model in Fig. 5 can be used to calculate the reliability R(t) of the 1-out-of-2 system under study.

$$R(t) = P_{START}(t) + P_{ONE-FAIL}(t) \tag{4}$$

where $P_{START}(t)$ is the probability of being in state START at time t and $P_{ONE-FAIL}(t)$ is the probability of being in state ONE-FAIL at time t. The model can also be used to obtain the Mean Time To Failure ($MTTF_{ft}$) of the system. $MTTF_{ft}$ can be calculated as follows (Billinton, 1983): First, the Stochastic Transitional Probability Matrix P for the model in Fig. 5 is obtained:

$$P = \begin{bmatrix} 1-(A+B) & A & B \\ D & 1-(D+E) & E \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where element p_{ij} is the transition rate from state *i* to state *j*. So, for example, p_{01} is equal to A=2cλ as in Fig. 5. But state LINE-FAIL is an absorbing state. Consequently, the truncated matrix Q is obtained from P by removing the rightmost column and the bottom row. So,

$$Q = \begin{bmatrix} 1-(A+B) & A \\ D & 1-(D+E) \end{bmatrix} \tag{6}$$

Let matrix M = $[I-Q]^{-1}$

$$M = \begin{bmatrix} (D+E)/L & A/L \\ D/L & (A+B)/L \end{bmatrix} \tag{7}$$

where L = {(A+B)(D+E)}- AD. M is generally defined as the fundamental matrix in which element m_{ij} is the average time spent in state *j* given that the system starts in state *i* before being absorbed. Since the system under study starts in state START and is absorbed in state LINE-FAIL,

$$MTTF_{ft} = m_{00} + m_{01} \tag{8}$$

For the system under study in this research,

$$MTTF_{ft} = \frac{A + D + E}{BE + BD + AE} \tag{9}$$

Expanding again in terms of λ, μ and c:

$$MTTF_{ft} = \frac{\lambda + \mu + 2c\lambda}{[(2\lambda)(1-c)][(\lambda+\mu] + [(2c\lambda)(\lambda)]} \tag{10}$$

6.2 Improving MTTF – First Approach

This section shows how to use the Markov model to improve system MTTF in a cost-effective manner. Let the 2-machine fault-tolerant production line described above, have the following parameters:

λ_1: controller failure rate
μ_1: controller repair rate
c_1: coverage

Increasing MTTF can be achieved by decreasing λ_1, increasing μ_1, increasing c_1 or a combination of the above. A possible answer to this question can be obtained by using operations research techniques in order to obtain a triplet ($\lambda_{optimal}$, $c_{optimal}$, $\mu_{optimal}$) that will lead to the highest MTTF. Practically, however, it may not be possible to find a controller with the exact failure rate $\lambda_{optimal}$ and/or the coverage $c_{optimal}$. Also, it may be difficult to find a maintenance plan with $\mu_{optimal}$. Upon contacting the machine's manufacturer, the factory will be offered a few choices in terms of better software versions and/or better maintenance plans. Better software will improve λ and c; the maintenance plan will affect μ. As mentioned above, let the initial value of λ, μ and c be {λ_1, c_1, μ_1}. Better software will change these values to {λ_j, c_j, μ_1} for $2 \le j \le n$. Here, n is the number of more sophisticated software versions. Practically, n will be a small number. Changing the maintenance policy will change μ_1 to μ_k for $2 \le k \le m$. Again, m will be a small number. In summary, system parameters {λ_1, c_1, μ_1} can only be changed to a small number of alternate triplets {λ_j, c_j, μ_k}. If n=3 and m=2, for example, the number of scenarios that need to be studied is *(mn-1)*=5. Running the Markov model 5 times will produce 5 possible values for the improved MTTF. Each scenario will obviously have a cost associated with it. Let

$$\eta = \frac{MTTF_{improved} - MTTF_{old}}{cost}$$

$MTTF_{old}$ is obtained by plugging (λ_1, c_1, μ_1) in the Markov model while $MTTF_{improved}$ is obtained using one of the other 5 triplets. η represents the improvement in system MTTF with respect to cost. The triplet that produces the highest η is chosen.

6.3 Improving MTTF – Second Approach

In this more complex approach, it is shown that λ, μ and c are not totally independent of each other. Let $Q_{software}$ be the quality of the software installed on the controller and let

$Q_{operator}$ represent the operator's expertise. A better version of the software (higher $Q_{software}$) will affect all three parameters simultaneously. Obviously, a better version of the software will have a lower software failure rate, thereby lowering λ. Furthermore, this better version is expected to have more sophisticated error detection and recovery mechanisms. This will increase the coverage c. Finally, the diagnostics capabilities of the software should be enhanced in this better version. This will reduce troubleshooting time, decrease the Repair time and increase μ.

Another important factor is the operator's expertise $Q_{operator}$. The controller is usually an industrial PC (Daoud et al., 2003). The machine manufacturer may be able to supply the hardware and software failure rates but the operator's expertise has to be factored in the calculation of the controller's failure rate on site. The operator does not just use the controller to operate the machine but also uses it for HTTP, FTP, e-mail, etc, beneficiating of its capabilities as a PC. Operator errors (due to lack of experience) will increase the controller failure rate. An experienced operator will make less mistakes while operating the machines. Hence, λ will decrease. Furthermore, an experienced operator will require less time to repair a controller, i.e., μ will increase.

In summary, an increase in $Q_{software}$ produces a decrease in λ and an increase in c and μ. Also, an increase in $Q_{operator}$ reduces λ and increases μ. Next, it is shown how to use $Q_{software}$ and $Q_{operator}$ to calculate λ, μ and c. The parameter λ can now be written as follows:

$$\lambda = \lambda_{hardware} + \lambda_{software} + \lambda_{operator} \tag{11}$$

The manufacturer determines $\lambda_{hardware}$. In general, let $\lambda_{software} = f(Q_{software})$. The function f is determined by the manufacturer. Alternatively, the manufacturer could just have a table indicating the software failure rate for each of the software versions. Similarly, let $\lambda_{operator} = g(Q_{operator})$. The function g has to be determined on site. Regarding the repair rate and the coverage, remember that, for an exponentially-distributed repair time, μ will be the inverse of the Mean Time To Repair (MTTR). There are two cases to be considered here. First, the factory does not stock controller spare parts on premises. Upon the occurrence of a controller failure, the agent of the machine manufacturer imports the appropriate spare part. A technician may also be needed to install this part. Several factors may therefore affect the MTTR including the availability of the spare part in the manufacturer's warehouse, customs, etc. Customs may seriously affect the MTTR in the case of developing countries, for example; in this case the MTTR will be in the order of two weeks. In summary, if the factory does not stock spare parts on site, the MTTR will be dominated by travel time, customs, etc. The effects of $Q_{software}$ and $Q_{operator}$ can be neglected.

Second, the factory does stock spare parts on site. If a local technician can handle the problem, the repair time should be just several hours. However, this does depend on the quality of the software and on the expertise of the technician. The better the diagnostic capabilities of the software, the quicker it will take to locate the faulty component. On the other hand, if the software cannot easily pinpoint the faulty component, the expertise of the technician will be essential to quickly fix the problem. If a foreign technician is needed, travel time has to be included in the repair time which will not be in the orders of several hours anymore. Let

$$\mu = P\{foreign - tech\}\mu_{foreign} + (1 - P\{foreign - tech\})\mu_{local} \tag{12}$$

μ_{local} is the expected repair rate in case the failure is repaired locally. μ_{local} is obviously a function of $Q_{software}$ and $Q_{operator}$. Let $\mu_{local} = h(Q_{software}, Q_{operator})$. The function h has to be determined on site. If a foreign technician is required, travel time and the technician's availability have to be taken into account. Again, here, the travel time is expected to dominate the actual repair time on site; in other words, the effects of $Q_{software}$ and $Q_{operator}$ can be neglected. The probability of requiring a foreign technician to repair a failure can be calculated as a first approximation from the number of times a foreign technician was required in the near past. The coverage parameter c has to be determined by the machine manufacturer.

Finally, to calculate the MTTF, the options are not numerous. The production manager will only have a few options to choose from. This approach is obviously more difficult to implement than the previous one. The determination of the functions *f, g* and *h* is not an easy task. On the other hand, using these functions permits the incorporation of the effect of software quality and operator expertise on λ, c and μ. The Markov model is used again to determine the MTTF for each triplet (λ, c, μ) and η determines the most cost-effective scenario. More details can be found in (Amer & Daoud 2006b).

7. Modeling Repair and Calculating Average Speed

The Markov chain in Fig. 5 has an absorbing state, namely state LINE-FAIL. In order to calculate system availability, the Markov chain should not have any absorbing states. System instantaneous availability is defined as the probability that the system is functioning properly at a certain time t. Conventional 1-out-of-2 Markov models usually model the repair as a transition from state ONE-FAIL to state START with a rate μ and another transition from state LINE-FAIL to state ONE-FAIL with a rate of 2μ (assuming that there are two repair persons available) (Siewiorek & Swarz, 1998). If there is only one repair person available (which is the realistic assumption in the context of developing countries), the transition rate from state LINE-FAIL to state ONE-FAIL is equal to μ. Figure 6 is the same Markov model as in Fig. 5 except for the extra transition from state LINE-FAIL back to state START. This model has a better representation of the repair policies in developing countries. In this improved model, the transition from state LINE-FAIL to state ONE-FAIL is cancelled. This is more realistic, although unconventional. Since most of the repair time is really travel time (time to import spare parts or time for a specialist to travel to the site), the difference in the time to repair one controller or two controllers will be minimal. In this model, the unavailability is equal to the probability of being in state LINE-FAIL while the availability is equal to the sum of the probabilities of being in states START and ONE-FAIL. These probabilities are going to be used next to calculate the average operating speed of the production line.

In (Daoud et al., 2005), it was found that a fully operational fault-tolerant production line with two machines can operate at a speed of 1.4S where S is the normal speed (1 revolution per minute as mentioned above). If one controller fails, the other controller takes charge of its duties and communicates with all sensors and actuators on both machines. The maximum speed of operation in this case was 1.3S. Assuming λ is not affected by machine speed, the average steady state speed ***Speed_Av$_{ss}$*** will be equal to:

$$Speed\_Av_{ss} = (P_{STARTss})\,(1.4S) + (P_{ONE-FAILss})\,(1.3S) \tag{13}$$

where $P_{STARTss}$ and $P_{ONE\text{-}FAILss}$ are the steady state probabilities of being in states START and ONE-FAIL respectively. If the machines had been operated at normal speed,

$$Speed\_Av_{ss} = \left(P_{STARTss}\right)\left(S\right) + \left(P_{ONE-FAILss}\right)\left(S\right) \tag{14}$$

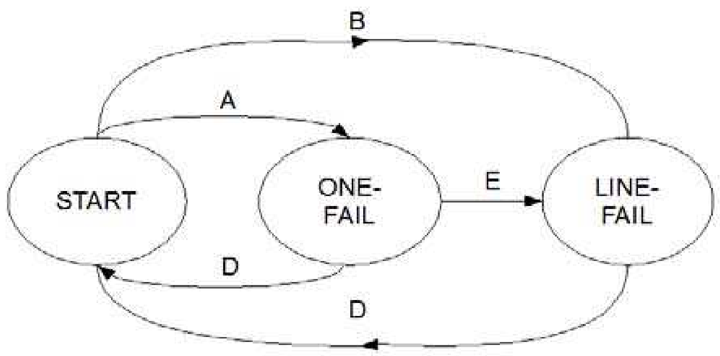


Fig. 6. Improved Markov model

Equations 13 and 14 can be used to estimate the increase in production when the machines are operated at higher-than-normal speeds. It is important to note here that machines are not usually operated at their maximum speed on a regular basis but only from time to time in order to obtain a higher turn-over. More information regarding this topic can be found in (Amer et al., 2005).

8. TMR Sensors

In the production line studied above, the sensors, switches and actuators were single points of failure. Introducing redundancy at the controller level may not be enough if the failure rate of the sensors/switches/actuators is relatively high especially since there are 32 sensors, 8 actuators, 3 switches and just two controllers. Introducing fault tolerance at the sensor level will certainly increase reliability. Triple Modular Redundancy (TMR) is a well-known fault tolerance technique (Johnson, 1989; Siewiorek & Swarz, 1998). Each sensor is triplicated. The three identical sensors send the same data to the controller. The controller compares the data; if the three messages are within the permissible tolerance range, the message is processed. If one of the three messages is different than the other two, it is concluded that the sensor responsible for sending this message has failed and its data is discarded. One of the other two identical messages is processed. This is known as masking redundancy (Johnson, 1989; Siewiorek & Swarz, 1998). The system does not fail even though one of its components is no longer operational. Triplicating each sensor in a light-traffic machine means that the machine will have 48 (=16*3) sensors, one controller and 4 actuators. The first important consequence of this extra hardware is the increased traffic on the network. The number of packets produced by sensors will be tripled. A machine with 48 sensors, one controller and 4 actuators was simulated and studied (Daoud et al. 2003); this is the *heavy-traffic* machine. The OPNET simulations in (Daoud et al., 2003) indicated that Gigabit Ethernet was able to accommodate both control and communication loads. Another important issue regarding the triplication of the sensors is cost-effectiveness. From a reliability point of view, triplicating sensors is expected to increase the system Mean Time

Between Failures (MTBF) and consequently, decrease the down time. However, the cost of adding fault tolerance has to be taken into account. This cost includes the extra sensors, the wiring, bigger switches and software modifications. The software is now required to handle the "voting" process; the messages from each three identical sensors have to be compared. If the three messages are within permissible tolerance ranges, one message is processed. If one of the messages is different from the other two, one of the two valid messages is used. The sensor that sent the corrupted message is disregarded till being repaired. If a second sensor from this group fails, the software will not be able to detect which of the sensors has failed and the production line has to be stopped. It is the software's responsibility to alert the operator using Human Machine Interface (HMI) about the location of the first malfunctioning sensor and to stop the production line upon the failure of the second sensor. System reliability is investigated next in order to find out whether or not the extra cost is justified.

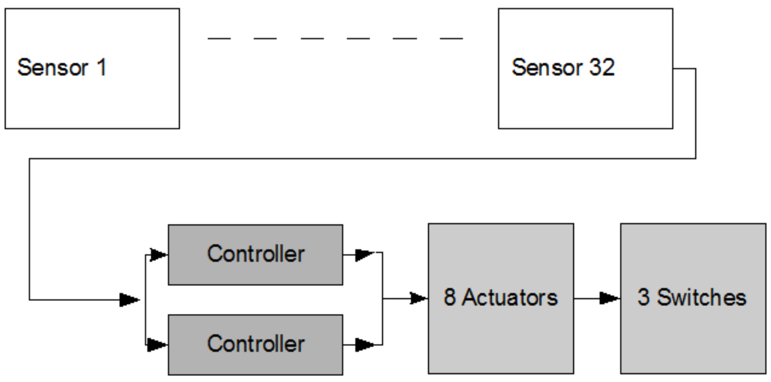


Fig. 7. RBD for Two-Cont Configuration

Reliability Block Diagrams (RBDs) can be used to calculate system reliability (Siewiorek & Swarz, 1998). Three configurations will be studied and compared. In the first configuration, there is no fault tolerance. Any sensor, controller, switch or actuator on either machine is a single point of failure. For exponentially-distributed failure times, the system failure rate is the sum of the failure rates of all its components. Let this configuration be the *Simplex* configuration. If fault tolerance is introduced at the controller level only (as in (Daoud et al., 2004b)), this configuration will be called *Two-Cont*. Figure 7 shows the RBD of the Two-Cont production line with two light-traffic machines. It is clear that fault tolerance only exists at the controller level. Figure 8 describes the RBD of the same production line but with two heavy-traffic machines. Now, every sensor is a TMR system and will fail when two of its sensors fail (2/3 system). Let this configuration be called the *TMR* configuration. Only the actuators and the switches constitute single points of failure. Instead of calculating system reliability, another approach is taken here, namely the Mission Time (**MT**). $MT(r_{min})$ is the time at which system reliability falls below r_{min} (Johnson, 1989; Siewiorek & Swarz, 1998). r_{min} is determined by production management and represents the minimum acceptable reliability for the production line. The production line will run continuously for a period of MT. Maintenance will then be performed; if one of the controllers has failed, it is repaired as well as any failed sensor. r_{min} is chosen such that the probability of having a system failure during MT is minimal.

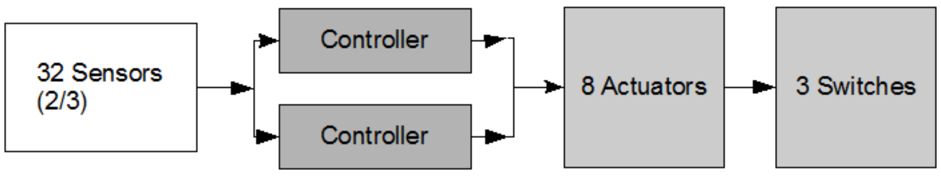


Fig. 8. RBD for TMR Configuration

It is assumed here that the production line is totally fault-free after maintenance. If r_{min} is high enough, there will be no unscheduled down time and no loss of production. Of course, if r_{min} is very high, MT will decrease and the down time will increase. Production can of course be directly related to cost. Let R_{line} be the reliability of the production line. R_{sensor}, R_{switch}, $R_{controller}$ and $R_{actuator}$ will be the reliabilities of the sensor, switch, controller and actuator, respectively. For exponentially-distributed failure times: $R = e^{-\lambda t}$. R is the component reliability (sensor, controller, ...) and λ is its failure rate (which is constant (Johnson, 1989; Siewiorek & Swarz, 1998)). Assume for simplicity that the switches are very reliable when compared to the sensors, actuators or controllers and that their probability of failure can be neglected. Furthermore, assume that all sensors on both machines have an identical reliability. The same applies for the controllers and the actuators. Next, the reliabilities of the production line will be calculated for the three configurations: Simplex, Two-Cont and TMR.

In the Simplex mode, there is no fault tolerance at all and any sensor, controller or actuator failure causes a system failure. Hence:

$$R_{line} = (R_{sensor}^{32})(R_{controller}^{2})(R_{actuator}^{8}) \tag{15}$$

Remember that each machine has 16 sensors, one controller and 4 actuators and the system (production line) consists of two machines. If fault tolerance is introduced at the controller level (as in (Daoud et al., 2004b)

$$R_{line} = \left(R_{sensor}^{32}\right)\left(1-(1-R_{controller})^{2}\right)\left(R_{actuator}^{8}\right) \tag{16}$$

The next level of fault tolerance is the introduction of Triple Modular Redundancy at the sensor level. Each of the 32 sensors will now be a sensor assembly that consists of three identical sensors. Hence

$$R_{line} = \left(3R_{sensor}^{2} - 2R_{sensor}^{3}\right)^{32}\left(1-(1-R_{controller})^{2}\right)\left(R_{actuator}^{8}\right) \tag{17}$$

Equations 15, 16 and 17 are then used to determine MT for a specific value of R_{line} for each of the three configurations. Hence, the cost-effectiveness of the added fault-tolerance can be quantitatively examined. More details can be found in (Amer & Daoud, 2008).

9. Conclusion

This chapter has discussed the performance and reliability of fault-tolerant Ethernet Networked Control Systems. The use of Gigabit Ethernet in networked control systems was investigated using the OPNET simulator. Real-time traffic and non-real time traffic were integrated without changing the IEEE 802.3 protocol packet format. In a mixed traffic industrial environment, it was found that standard Gigabit Ethernet switches succeeded in meeting the required time constraints. The maximum speed of operation of individual machines and fault tolerant production-lines was also studied.
The reliability and availability of fault tolerant production lines was addressed next. It was shown how to use Markov models to find the most cost-effective way of increasing the Mean Time To Failure MTTF. Improved techniques for modeling repair were also discussed. Finally, it was shown how to introduce fault tolerance at the sensor level in order to increase production line mission time.

10. References

Amer, H.H. & McCluskey, E.J. (1986). "Calculation of the Coverage Parameter for the Reliability Modeling of Fault-tolerant Computer Systems", *Proc. Intern. Symp. on Circuits and Systems ISCAS*, pp. 1050-1053, San Jose, CA, U.S.A., May 1986.

Amer, H.H. & McCluskey, E.J. (1987a). "Weighted Coverage in Fault-tolerant Systems", *Proc. Reliability and Maintainability Symp. RAMS*, pp.187-191, Philadelphia, PA, U.S.A., January 1987.

Amer, H.H. & McCluskey, E.J. (1987b). "Latent Failures and Coverage in Fault-tolerant Systems", *Proc. Phoenix Conf. on Computers and Communications*, Scottsdale, pp. 89-93, AZ, U.S.A., February 1987.

Amer, H.H. & McCluskey, E.J. (1987c). "Calculation of Coverage Parameter", *IEEE Trans. Reliability*, June 1987, pp. 194-198.

Amer, H.H.; Moustafa, M.S. & Daoud, R.M. (2005). “Optimum Machine Performance In Fault-Tolerant Networked Control Systems”, *Proceedings of the IEEE EUROCON Conference*, pp. 346-349, Belgrade, Serbia & Montenegro, November 2005.

Amer, H.H.; Moustafa, M.S. & Daoud, R.M. (2006a). “Availability Of Pyramid Industrial Networks”, *Proceedings of the Canadian Conference on Electrical and Computer Engineering CCECE*, pp. 1862-1865, Ottawa, Canada, May 2006.

Amer, H.H. & Daoud, R.M. (2006b). “Parameter Determination for the Markov Modeling of Two-Machine Production Lines” *Proceedings of the International IEEE Conference on Industrial Informatics INDIN*, pp. 1178-1182, Singapore, August 2006.

Amer, H.H. & Daoud, R.M. (2008). “Increasing Network Reliability by Using Fault-Tolerant Sensors”, *International Journal of Factory Automation, Robotics and Soft Computing*, January 2008, pp. 71-76.

Arnold, T.F. (1973). “The concept of coverage and its effect on the reliability model of a repairable system,” *IEEE Trans. On Computers*, vol. C-22, No. 3, March 1973.

Baillieul, J. & Antsaklis, P.J. (2007). “Control and Communication Challenges in Networked Real-Time Systems”, *Proceedings of the IEEE*, Vol. 95, No. 1, January 2007, pp. 9-28.

Billinton, R. & Allan, R. (1983) “*Reliability Evaluation of Engineering Systems: Concepts and Techniques*”, Pitman.

Blanke, M.; Kinnaert, M.; Lunze, J. & Staroswiecki, M. (2006). *"Diagnosis and Fault-Tolerant Control",* Springer-Verlag.

Bossar Horizontal Machinery. Official Site: www.bossar.es

Brahimi, B.; Aubrun, C. & Rondeau, E. (2006). "Modelling and Simulation of Scheduling Policies Implemented in Ethernet Switch by Using Coloured Petri Nets," *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation ETFA*, Prague, Czech Republic, September 2006.

Brahimi, B. (2007). "Proposition d'une approche intégrée basée sur les réseaux de Petri de Haut Niveau pour simuler et évaluer les systèmes contrôlés en réseau," PhD Thesis, Université Henri Poincaré, Nancy I, December 2007.

Bushnell, L. (2001). "Networks and Control", *IEEE Control Systems Magazine*, vol. 21, no. 1, 2001, pp. 22-23.

Clauset, A., Tanner, H.G., Abdallah, C.T., & Byrne, R.H. (2008). "Controlling Across Complex Networks – Emerging Links Between Networks and Control", *Annual Reviews in Control* , Vol. 32, No. 2, pp. 183–192, December 2008.

ControlNet, Official Site: http://www.controlnet.org

Daoud, R.M.; Elsayed, H.M.; Amer, H.H. & Eid, S.Z. (2003). "Performance of Fast and Gigabit Ethernet in Networked Control Systems," *Proceedings of the IEEE International Mid-West Symposium on Circuits and Systems, MWSCAS*, Cairo, Egypt, December 2003.

Daoud, R.M. (2004a). *Performance of Gigabit Ethernet in Networked Control Systems*, MSc Thesis, Electronics and Communications Department, Faculty of Engineering, Cairo University, 2004.

Daoud, R.M.; Elsayed, H.M. & Amer, H.H. (2004b). "Gigabit Ethernet for Redundant Networked Control Systems, *Proceedings of the IEEE International Conference on Industrial Technology ICIT*, December 2004, Hammamet, Tunis.

Daoud, R.M., Amer, H.H. & Elsayed, H.M. (2005). "Fault-Tolerant Networked Control Systems under Varying Load," *IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications, SMCia,* Espoo, Finland, June 2005.

Daoud, R.M. & Amer, H.H. (2007). "Ethernet for Heavy Traffic Networked Control Systems", *International Journal of Factory Automation, Robotics and Soft Computing,* January 2007, pp. 34-39.

Daoud, R.M. (2008). *Wireless and Wired Ethernet for Intelligent Transportation Systems*, DSc Dissertation, LAMIH-SP, Universite de Valenciennes et du Hainaut Cambresis, France, 2008.

Decotignie, J.-D. (2005). "Ethernet-Based Real-Time and Industrial Communications," *Proceedings of the IEEE*, vol. 93, No. 6, June 2005.

Eker, J. & Cervin, A. (1999). "A Matlab Toolbox for Real-Time and Control Systems Co-Design," *6th International Conference on Real-Time Computing Systems and Applications*, Hong Kong, P.R. China, December 1999.

EtherNet/IP Performance and Application Guide, Allen-Bradley, Rockwell Automation, Application Solution.

Felser, M. (2005). "Real-Time Ethernet – Industry Prospective," *Proceedings of the IEEE*, vol. 93, No. 6, June 2005.

Georges, J.-P. (2005). "Systèmes contrôles en réseau: Evaluation de performances d'architectures Ethernet commutées," PhD thesis, Centre de Recherche en Automatique de Nancy CRAN, November 2005.

Georges, J.P.; Vatanski, N.; Rondeau, E. & Jämsä-Jounela, S.-L. (2006). "Use of Upper Bound Delay Estimate in Stability Analysis and Robust Control Compensation in Networked Control Systems," *12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM*, St-Etienne, France, May 2006.

Grieu, J. (2004). "Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques," PhD Thesis, Institut National Polytechnique de Toulouse, Ecole doctorale informatique et telecommunications, September 2004.

IEEE Std 802.3, 2000 Edition

Jasperneite, J. & Elsayed, E. (2004). "Investigations on a Distributed Time-triggered Ethernet Realtime Protocol used by PROFINET," *3rd International Workshop on Real-Time Networks (RTN 2004)*, Catania, Sicily, Italy , Jun 2004.

Johnson, B. W. (1989). "*Design and Analysis of Fault-Tolerant Digital Systems*", Addison-Wesley.

Hespanha, J.P. , Naghshtabrizi, P. & Xu, Y (2007). "A Survey of Recent Results in Networked Control Systems", *Proceedings of the IEEE,* Vol. 95, No. 1, January 2007, pp. 138-162.

Kumar, P.R. (2001). "New Technological Vistas for Systems and Control: The Example of Wireless Networks," *IEEE Control Systems Magazine*, vol. 21, no. 1, 2001, pp. 24-37.

Lee, S.-H. & Cho, K.-H. (2001). "Congestion Control of High-Speed Gigabit-Ethernet Networks for Industrial Applications," *Proc. IEEE ISIE,* Pusan, Korea, pp. 260-265, June 2001.

Lian, F.L.; Moyne, J.R. & Tilbury, D.M. (1999). "Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet," Tech. Rep. UM-MEAM-99-02, February 1999. Available: http://www.eecs.umich.edu/~impact

Lian, F.L.; Moyne, J.R. & Tilbury, D.M. (2001a). "Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet," *IEEE Control Systems Magazine*, Vol. 21, No. 1, pp.66-83, February 2001.

Lian, F.L.; Moyne, J.R. & Tilbury, D.M. (2001b). "Networked Control Systems Toolkit: A Simulation Package for Analysis and Design of Control Systems with Network Communication," Tech. Rep., UM-ME-01-04, July 2001.
Available: http://www.eecs.umich.edu/~impact

Lounsbury, B. & Westerman, J. (2001). "Ethernet: Surviving the Manufacturing and Industrial Environment," Allen-Bradley white paper, May 2001.

Marsal, G. (2006a). "Evaluation of time performances of Ethernet-based Automation Systems by simulation of High-level Petri Nets," *PhD Thesis,* Ecole Normale Superieure De Cachan, December 2006.

Marsal, G.; Denis, B.; Faur, J.-M. & Frey, G. (2006b). "Evaluation of Response Time in Ethernet-based Automation Systems," *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Prague, Czech Republic, September 2006, pp. 380-387.

Meditch, J.S. & Lea, C.-T. (1983). "Stability and Optimization of the CSMA and CSMA/CD Channels," *IEEE Trans. Comm.*, Vol. 31, No. 6 , June 1983, pp. 763-774.

Morriss, S.B. (1995). "*Automated Manufacturing Systems Actuators, Controls, Sensors, and Robotics"*, McGraw-Hill.

Nilsson, J., "Real-Time Control Systems with Delays," PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.

ODVA, "Volume 1: CIP Common," Available: http://www.odva.org/10_2/03_events/03_ethernet-homepage.htm

ODVA, "Volume 2: EtherNet/IP Adaptation on CIP," Available: http://www.odva.org/10_2/03_events/03_ethernet-homepage.htm

Opnet, Official Site for OPNET http://opnet.com

Siewiorek, D.P. & Swarz, R.S. (1998). "*Reliable Computer Systems – Design and Evaluation,*" A K Peters, Natick, Massachusetts.

Skeie, T.; Johannessen, S. & Brunner, C. (2002). "Ethernet in Substation Automation," *IEEE Control Syst*., Vol. 22, no. 3, June 2002, pp. 43-51.

Soloman, S. (1994). "*Sensors and Control Systems in Manufacturing*," McGraw-Hill.

Sundararaman, B.; Buy, U. & Kshemkalyani, A.D. (2005). "Clock Synchronization for Wireless Sensor Networks: a survey," Ad Hoc Networks, vol. 3, 2005, pp. 281-323.

Thomesse, J.-P. (2005). "Fieldbus Technology in Industrial Automation", *Proceedings of the IEEE,* Vol. 93, No. 6, June 2005, pp. 1073-1101.

Tolly, K. (1997). "*The Great Networking Correction: Frames Reaffirmed,*" Industry Report, The Tolly Group, IEEE Internet Computing, 1997.

Trivedi, K.S. (2002). *"Probability and Statistics with Reliability, Queuing, and Computer Science Applications",* Wiley, New York.

Vatanski, N.; Georges, J.P.; Aubrun, C.; Rondeau, E. & Jämsä-Jounela, S.-L. (2006). "Control Compensation Based on Upper Bound Delay in Networked Control Systems," *17th International Symposium on Mathematical Theory of Networks and Systems, MTNS*, Kyoto, Japan, July 2006.

Walsh, G.C. & Ye, H. (2001). "Scheduling of Networked Control Systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, February 2001, pp. 57-65.

Wang, J. & Keshav, S. (1999). "Efficient and Accurate Ethernet Simulation," Cornell Network Research Group (C/NRG), Department of Computer Science, Cornell University, May 1999.

Wittenmark, B.; Bastian, B. & Nilsson, J. (1998). "Analysis of Time Delays in Synchronous and Asynchronous Control Loops," Lund Institute of Technology, 37th CDC, Tampa, December 1998.

Yang, T.C. (2006). "Networked Control System: a Brief Survey", *IEE Proceedings-Control Theory and Applications.*, Vol. 153, No. 4, July 2006, pp. 403-412.

Zhang, W.; Branicky, M.S. & Phillips, S.M. (2001). "Stability of Networked Control Systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, February 2001, pp. 84-99.