6-1-2014

Performability of Integrated Networked Control Systems

Eslam Moustafa

The American University in Cairo

School of Sciences and Engineering

PERFORMABILITY OF INTEGRATED NETWORKED CONTROL SYSTEMS

A Thesis Submitted to

The Electronics Engineering Department

in partial fulfillment of the requirements for
the degree of Master of Science

by Eslam Abd Elatif Moustafa Abd Elatif

under the supervision of Prof. Hassanein H. Amer and Dr. Ramez M. Daoud
May/2014

DEDICATION

I would dedicate this thesis to my dear family. My parents and my sisters who have been always supporting me in my life. I would also like to dedicate it to all my professors.

Mom: I cannot imagine my life without you. You help and support me all the time and I owe you all the success in my life.

Dad: Like Father Like Son. You are my role model and without your help and faith, I could not be the man I am today.

I would like to especially dedicate this thesis to Prof. Hassanein Amer and Dr. Ramez Daoud. Prof. Hassanein Amer has been my favorite professor since I was an undergraduate student as we both have white hearts. I wish if I did my undergraduate thesis with you and my wish came true by doing my masters under your supervision. Whatever I did, I wouldn't be able to pay back one of the favors you did for me. I wish you all success in your academic and personal life. I have also learnt a lot from Dr. Ramez. You are such an inspiration to me. I would like to thank both of you for all the efforts you exerted teaching and supervising me. It has been an honor working with both of you and I hope that this is not the end of our cooperation. One Day, I will be back.

Moreover, I would like to thank all my friends for their help and support.

May Allah grant you all happiness in this life.

ACKNOWLEDGMENTS

I would like to acknowledge both my supervisors: Prof. H.H. Amer and Dr. R.M. Daoud for their endless support throughout this thesis.

I would also like to acknowledge Eng. Hassan Halawa for his generous assistance and support throughout my work.

I would finally like to acknowledge my examiners: Prof. Magdy El Soudani and Prof. Ahmed Abou-Auf and the graduate program directors: Dr. Ayman El Ezabei and Dr. Karim Seddik.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NCS	Networked Control System
S2A	Sensor To Actuator
SANET	Sensor / Actuator Network
CAN	Controller Area Network
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
BEB	Binary Exponential Backoff
TT	Time-Triggered
FTT	Flexible TT
CIP	Common Industrial Protocol
UDP	User Datagram Protocol
FTP	File Transfer protocol
HTTP	Hypertext Transfer Protocol
T-S	Takagi-Sugeno
TP	Transient Performability
CTMC	Continuous Time Markov Chain
TCP	Transmission Control Protocol
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair

LIST OF SYMBOLS

h	Sampling Period
τsc	Sensor To Controller Delay
τc	Controller Computation Delay
τca	Controller To Actuator Delay
D	Delay
L	Packet Length
R	Link Transmission Rate
d	Link Length
s	Propagation Speed
λ	Failure Rate
μ	Repair Rate
c	Coverage
R	Reliability

ABSTRACT

The American University in Cairo, Egypt

Performability of Integrated Networked Control Systems

Name: Eslam Abd Elatif Moustafa Abd Elatif

Supervisors: Prof. Hassanein H. Amer and Dr. Ramez M. Daoud

A direct sensor to actuator communication model (S2A) for unmodified Ethernet-based Networked Control Systems (NCSs) is presented in this research. A comparison is made between the S2A model and a previously introduced model including an in-loop controller node. OMNET simulations showed the success of the S2A model in meeting system delay with strict zero packet loss (with no over-delayed packets) requirements. The S2A model also showed a reduction in the end-to-end delay of control packets from sensor nodes to actuator nodes in both Fast and Gigabit switched Ethernet-Based. Another major improvement for the S2A model is accommodating the increase in the amount of additional load compared to the in-loop model.

Two different controller-level fault-tolerant models for Ethernet-based Networked Control Systems (NCSs) are also presented in this research. These models are studied using unmodified Fast and Gigabit Ethernet. The first is an in-loop fault-tolerant controller model while the second is a fault-tolerant direct Sensor to Actuator (S2A) model. Both models were shown via OMNeT++ simulations to succeed in meeting system end-to-end delay with strict zero packet loss (with no over-delayed packets) requirements. Although, it was shown that the S2A model has a lower end-to-end delay than the in-loop controller model, the fault-tolerant in-loop model performs better than the fault-tolerant S2A model in terms of less total end-to-end delay in the fault-free situation. While, on the other hand, in the scenario with the failed controller(s), the S2A model was shown to have less total end-to-end delay.

Performability analysis between the two fault-tolerant models is studied and compared using fast Ethernet links relating controller failure with reward, depending on the system state. Meeting control system's deadline is essential in Networked Control Systems and failing to meet this deadline represents a failure of the system. Therefore, the reward is considered to be how far is the total end-to-end delay in each state in each model from the system deadline. A case study is presented that simultaneously investigates the failure on the controller level with reward.

I. INTRODUCTION

There are two types of networks either control networks or data communication networks. Manufacturing control has been moving more and more towards distributed implementations of control systems. Networks are used to communicate the data, instead of using traditional point-to-point communication. Networks require less wiring and less maintenance compared to a point-to-point architecture. Such networks carry a large number of small control signals between many nodes and these signals have to meet the delay constraints of real-time control systems. The main difference between such control networks and conventional data networks is that control networks must be able to support time-critical applications. Networked control systems share certain aspects across the range of different applications.

An NCS is composed of Sensors (S), Actuators (A) and a Controller (K). Sensors, controllers and actuators communicate together over a network. Sensors send packets to the controllers which calculate the control action that should be delivered to the actuators, and these transmissions must meet the control system's deadline as shown in Figure 1.

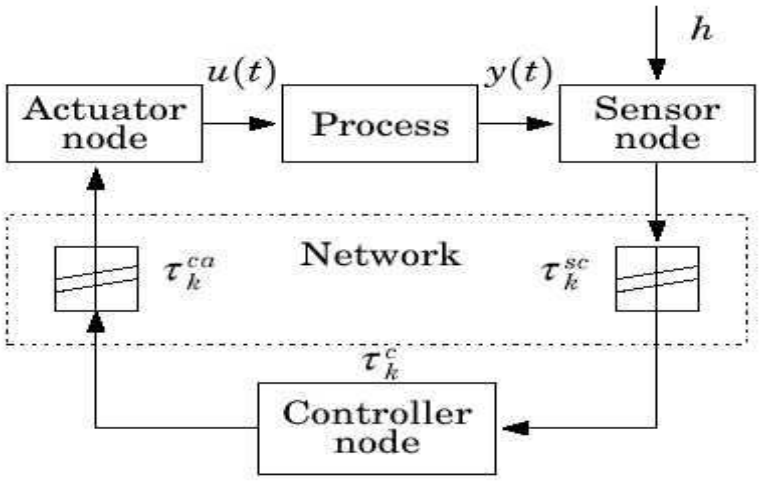

Figure 1: Networked Control System

There are four factors that affect the utilization of the network bandwidth: the sampling rate, the number of nodes requiring synchronous operation, the size of the information sent and the protocol used. There are two types of NCS systems either time-triggered (or clock-driven) or event-triggered. A clock-driven system consists of sensors and actuators (SAs) with constant sampling periods where samples are taken at discrete time points. On the other hand, an event-triggered system has continuous sampling and an event triggers the control process. The time taken by a packet to travel from S to K and K to A respectively is considered the total end-to-end delay which includes all types of encapsulation/decapsulation, propagation and queuing delays.

In this research, a direct sensor to actuator communication model (S2A) for Ethernet-based Networked Control Systems (NCSs) is presented in this research where a comparison is made between the S2A model and a previously introduced model including an in-loop controller node. Then, two different controller-level fault-tolerant models for Ethernet-based Networked Control Systems (NCSs) are also presented in this research. These models are studied using unmodified Fast and Gigabit Ethernet. Finally, a performability analysis for the two fault-tolerant models is studied and compared using fast Ethernet links relating controller failure with reward and a case study is presented at the end.

Chapter II summaries the literature review. First, the use of Ethernet (IEEE 802.3) in the context of NCS is illustrated. Then, performance of Fast and Gigabit Ethernet in Networked Control Systems is studied. Then, direct sensor actuator integrated approach is proposed. Finally, fault-tolerance techniques and performability models are presented.

In Chapter III, new direct sensor to actuator (S2A) architecture is developed; it has 16 sensors and 4 actuators. However, each sensor communicates with the appropriate actuator(s) directly without going through a controller node. In other words, each actuator incorporates its own control function as in. This proposed architecture was studied on-top-of both Fast and Gigabit switched Ethernet. It was shown that this architecture succeeds in meeting the required time constraints. Then, the architecture was compared to a traditional in-loop controller architecture and it was shown, via OMNeT++ simulations, that the observed end-to-end delay is smaller in the proposed architecture. Finally, it was shown that the proposed model can withstand more additional load than a system with an in-loop controller as in.

In Chapter IV, new models are developed where the focus is on applying fault-tolerance techniques on the control level of both architectures. These fault-tolerance techniques will increase the reliability of the architectures as well as their lifetime. New models are developed on-top-of both Fast and Gigabit switched Ethernet. Comparison is made between the two proposed models via OMNeT++ simulations where the focus is on factors such as the number of packets dropped and the observed end-to-end delay. Finally, performability analysis of the two models is investigated in chapter V, where a case study is presented using practical numbers from the industry.

This thesis is concluded in Chapter VI.

II. LITERATURE REVIEW

Traditionally, for proper control, there are different protocols used which have a deterministic behavior such as DeviceNET and ControlNET [1, 2]. Also, many real-time applications were studied using protocols such as Controller Area Network (CAN), PROFlBUS and EtherNet/IP which is a merger between Ethernet and ControlNET [2-6]. However, with the natural demand for higher bandwidth and accessibility, more robust and non-deterministic protocols such as Ethernet made their way into the world of real-time NCS [7-13].

II.1 ETHERNET IN NCS

Ethernet has recently appeared in the world of wired communication systems, and the implementation of Ethernet as a communication medium for Networked Control System became important. Although Ethernet is a non-deterministic protocol by nature, researchers in academia and industry did not stop using the Ether-Channel as a communication medium for control systems. Because of the real-time constraints inherent in control systems, the non-deterministic nature of Ethernet is thought to be challenging; however, it was showed through research that Ethernet (or IEEE Std 802.3) can perform well in Networked Control System either by changing packet format for real-time control messages, or by giving higher priority for these messages [14-16].

Also, one of the sources of randomness in Ethernet which stood against its use for real-time NCS applications, is the utilization of Carrier Sense Multiple Access with Collision Detection (CSMA/CD) [17]. The concept of Binary Exponential Backoff (BEB) is implemented in this technique, where a transmitting node ‘backs off’ from transmission upon detection of a collision. The duration for this backoff is a

value between 0 and 2k – 1 time-slots, where k is the number of collisions detected/avoided. The backoff duration grows exponentially as the number of collisions increases. Therefore, in order to decrease the effect of this randomness, several modifications were made to the Ethernet standard, specifically to accommodate real-time applications. These modifications include (but not limited to) EtherNet/IP, Time-Triggered Ethernet (TT Ethernet) and Flexible TT Ethernet (FTT Ethernet) [5, 14-19]. Ethernet/IP was proposed by Rockwell Automation and the ODVA organization as an industrial version of Ethernet and they have developed the Common Industrial Protocol (CIP) [16, 20]. Recently unmodified Ethernet for use in real-time applications has been standardized [21, 22].

Also, it was shown, in many studies, that Ethernet can be used in NCSs [23-31]. The use of Ethernet without modification as a control protocol has also been studied in multiple researches [1, 9-12, 32-34].

With the use of Ethernet, 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. Therefore, 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. 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) can easily be done. Furthermore, some tasks can be enabled that like web-browsing and email 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 [25].

II.2 Performance of Fast and Gigabit Ethernet in Networked Control Systems

In [25], the use of Fast and Gigabit Ethernet in networked control systems was tested. Real-time traffic and non-real time traffic were integrated without changing the IEEE 802.3 protocol packet format. It was found in a mixed traffic industrial environment that standard Gigabit Ethernet switches succeed to meet time constraints while Fast Ethernet fail to meet. A simulation study of Ethernet networks that integrate real-time control packets with other communication packets was conducted. Various loading cases in both Fast and Gigabit Ethernet networks were considered to test the effect of increased network speed on NCS performance.

In early works such as [15], the medium access sub-layer of CSMA/CD was modified to distinguish between real-time and other traffic packets. Studies were conducted for testing the stability of the communication channel and optimizing its performance. In [7], Fast Ethernet was tested to eliminate incompatible communication networks at the traditional substation automation. Using Fast Ethernet was tested in the switched topology in power station control application. This study was done by ABB for economic and standardization reasons. The results of this study were satisfactory within the time frame of the considered application. Fast Ethernet switch topology succeeded to run this system because the application presented had relatively large time frame limit. Finally, in [14], contention over the Ethernet channels when used in control was studied at high speeds.

In [15], two models were built to study the performance of Fast and Gigabit Ethernet in Networked Control Systems. One model is run on top of Fast Ethernet and the other one is run over Gigabit Ethernet for performance comparison. The first model consists of 16 sensors, one controller, and 4 actuators, based on the model of

[7]. The first model is called the light traffic system. While the other model consists of 48 sensors, one controller, and 4 actuators, and it is called the heavy traffic system. A machine running at a speed of 1 revolution per second is encoded into 1,440 electric pulses for electrical synchronization and control over traditional PLCs [7]. Therefore, the sampling frequency is 1,440 Hz and the system will have a deadline of 694μs. In other words, a control action must be taken within a frame of 694μs as round-trip delay originating from the sensor, passing through the controller, and transmitted once more over the network to reach the actuator.

OPNET was used as a simulation platform where all packets were treated in the switch in a similar manner without prioritization. Therefore, the packet format of the IEEE 803.2z standard [8] was used without modification. Control signals in the simulations are UDP packets. Also, the packet size was fixed to minimum frame size in Gigabit Ethernet (520 bytes). The effect of mixing the control traffic with other types of traffic was considered during simulation where the option of on-line system diagnostic and fix-up (log-on, request/download file, up-load file, log-off) is included as well as e-mail and web-browsing. FTP of 101KB files was also considered, which represents small download/upload data [7]. Finally, HTTP, E-mail and telnet traffic was added using OPNET built-in heavy load models.

The simulation results showed that with high speed Ethernet networks, standard switches can accommodate the timing requirements of many control systems. Also, additional traffic resulting from integration of other functions did not affect the control packets, as long as this traffic is kept within reasonable limits.

II.3 DIRECT SENSOR ACTUATOR INTEGRATED APPROACH

In [35], it was shown that by a separate control design and its posterior distributed implementation, the system performance may suffer degradation. When control loops are closed over communication network, varying delays can appear and decrease the control system performance, and even lead the system to instability. However, it was showed that by an adequate integrated approach, the system performance increases dramatically.

A real-time distributed control system is typically implemented by a set of computational devices (sensors, actuators, controllers, etc). These devices run one or several tasks, which communicate data across a field level communication network (fieldbus). The successful design and implementation of real-time distributed control application requires an appropriate integration of several disciplines including control systems, real-time systems and communication systems. The key for distributed control systems is that almost no local control action can be taken in isolation from the rest of the system. Sampling, control computation, and actuation are the main parts of a control loop. According to control theory, sampling should be performed at the same instant every period, control computation should start and finish quickly after the sample is available, and finally actuation should occur immediately after the control computation, or at a fixed instant after the sampling depending the controller design. In control theory, the three main parts of a control loop are assumed to be instantaneous. However, when several field devices exchange data over fieldbus communication networks, at run time, control loop timing assumptions are not met due to timing problems, leading to violations that can cause degradation in control performance and even instability. The control loop is implemented in a distributed

architecture, with three nodes communicating across a fieldbus communication network, as shown in Figure 2 [35].

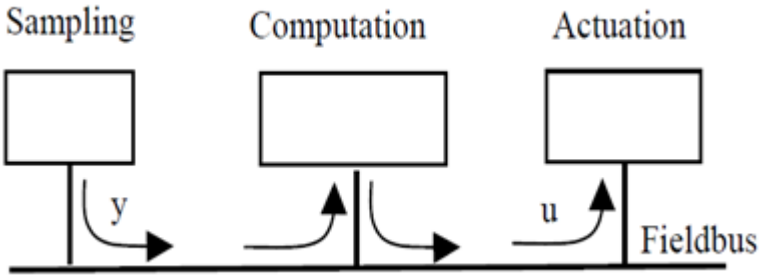

Figure 2: Fieldbus-based Distributed Architecture

A sensor node, strictly periodically (h, sampling period), samples the system (y(t)) and sends the data to the controller node, introducing a communication delay (τsc, sensor to controller delay). A controller node, that executes a single control computation, introduces a computation delay (τc), that is assumed to be constant for each controller execution. Finally, when the output is produced (u(t)), it is sent to the actuator node, introducing again another communication delay (τca, controller to actuator delay).

An integrated approach is proposed where the control computation is moved from the controller node to the actuator node and the controller node is removed as shown in Figure 3 [35].

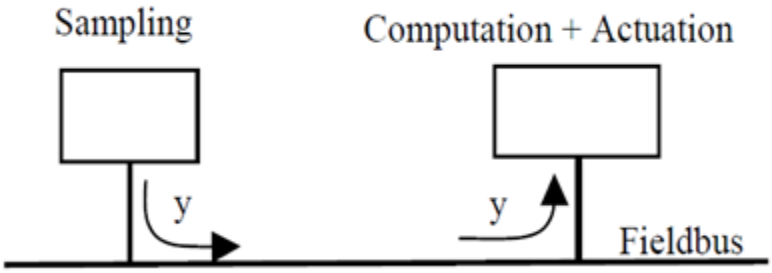

Figure 3: New Distributed Architecture

This same concept is implemented in Sensor Actuator Networks (SANETs) where a group of sensors and actuators are distributed geographically and communicate together through wired or wireless networks [36-38].

II.4 FAULT-TOLERANCE AND PERFORMABILITY

Different fault-tolerance techniques were applied previously on the node level (sensors, controllers, actuators) in Networked Control Systems. In [39], Triple Modular Redundancy (TMR) fault tolerance technique is studied. TMR can cover single faults in the system by replicating a block three times. The output of the three blocks enters into a voter where a majority voting process takes place to decide the correct output. This technique could be applied on the sensors level where there are three sensors connected to a voter and when two sensors have close readings while the third has a completely different one, the voter will choose the reading of the two sensors as opposed to the single sensor. Therefore, TMR is an excellent way to prevent a system failure due to a single event upset. However, if there is more than one fault in the system, TMR will not be able to perform its function because it will not be able to decide the correct output based on a majority vote. The reliability of TMR could be improved by using three voters instead of one voter which represents a single point of failure. However, improved TMR with three voters would be slightly slower than non-redundant circuit and would increase the cost because of the additional hardware resources.

In [40], redundant control node was used for connecting two machines for minimum down-time using unmodified Fast and Gigabit Ethernet. These two machines were operating with isolated controllers (one controller for each machine) and they were connected by the industrial floor network. When one of the controllers fails, its task must be shifted to the running machine through the operating controller.

The performance degradation of the system was studied upon the failure of one of the controllers. The main function of the controller of each machine is to take charge of machine control and to help in synchronization between the two machines. In order to achieve synchronization, a status vector is sent between the two controllers of the two machines. This status vector includes complete knowledge of machine information such as the cam position, the production rate, and so on. The machines can speed up or slow down to match their respective productions based on the status of the test vector. Also, the two controllers can back-up data on each other which will achieve fault-tolerance on the controller level. Although the production process can be slowed down, the production is not stopped. Another feature to enhance fault-tolerance is having a supervisory controller in order to monitor the status of the two machines. This supervisory controller takes over when one of the two controllers fails or even upon the failure of both of them. Three simulations were run using OPNET where the first scenario includes two machines working in line, while the second scenario includes a failed controller whose traffic is switched to the operating one. Finally, the last scenario includes two failed controllers on two machines in-line with a third functioning machine where the traffic of the two failed controllers is deviated to the third controller which increases the real time load. Simulation results showed that the delay is too large when using Fast Ethernet links while the delay was small for Gigabit Ethernet. It is concluded that Gigabit Ethernet can accommodate the real-time traffic and deliver packets within the required time frame compared to Fast Ethernet which fails in meeting the required system time constraints which represent a system failure.

In [41], the fault tolerant ability of networked machines is tested by reallocating loads in case of controller failure via OPNET simulations. Also, the

maximum speed of operation of individual machines and fault tolerant production-lines is studied. All machine networks are built on-top-of switched Gigabit Ethernet using Star topology. The simulations showed that the system can tolerate three failed controllers while connecting up to 4 machines on-top-of Gigabit Ethernet. It was also found that the network can absorb an increase in machine speed. It was also shown that upon the failure of all controllers except one, two-machine production line can tolerate an increase in speed greater than the increase a 3 and 4-machine production line can tolerate.

A supervisory control level is essential in many distributed control systems where the functions are hierarchal. The role of this level is monitoring the control objectives and supporting the overall coordinated control in different phases of normal operation. Also, this level allows the diagnosis of all foreseeable faults, takes the necessary corrective actions, including the change of controller parameter or structure [42].

In [43], a pyramid control hierarchy is proposed based on the presence of a supervisor controller on top of separate controller nodes where two models are tested. In the first model, there are one supervisor/two sub-controllers, while in the second model, there are one supervisor/three sub-controllers. All possible combinations of supervisor-controller inter-communication are tested where all supervisor/controller inter-changeability possibilities are taken into consideration. A simulation study is conducted to test the functionality of the system using switched Gigabit Ethernet in Star topology. Each model is built where running machines are connected for in-line production, and they are monitored by a supervisor controller. The supervisor controller is either passive or active. In normal operation, when all controllers are running with no production difficulties, the supervisor collects information from the

controllers it is mastering. It could be represented by a tree structure with the supervisor as the root and the controllers as the leaves. Inter-leaves communication takes place during in-line production scheme. In the passive mode, information is collected by the root and displayed on the main control room screen. In the active mode, the supervisor node turns on to be active taking over control of the machine with failed controller. Also, it can switch the control of the failed machine to another operating controller on the same network. The supervisor can also take over the control function upon the failure of all the controllers of the running machines. Two simulations scenarios were tested using OPNET. The first scenario focuses on two machine model with a supervisor while the second scenario focuses on three machine model with a supervisor. The results showed that best back-up scenario for the two machine model failed controller is to be replaced by the supervisor node. Also, it was shown that the best back-up scenario for three machine failed controller is to be replaced by the supervisor, not by one of its neighboring controllers in order to keep balanced traffic load among controllers. It is recommended that the supervisor have computational capacity double of any other controller it is supervising in order to be able to back-up two failed controllers and have successful communication with the remaining controller. Finally, note that upon the failure of the active supervisor, the entire system goes out of service because it is responsible for inter-machine controllers' communication.

In [44], the availability of the pyramid architecture in the context of Networked Control Systems is studied where two machines are working in an in-line production and supervised by an upper level node running on top of Gigabit Ethernet using star topology. It was also shown from a reliability point of view, the importance of having an access panel on at least one of the machines. Therefore, the supervisor

reliability has to be much higher than that of the machines. Markov models are used to calculate system availability and can also be used as a design tool. There are three different modes: MAX, PARTIAL, and MIN. In MAX mode, both of the machines and the supervisor have panels while in the PARTIAL mode, only one of the machines is equipped with a panel. In MIN mode, only the supervisor has a panel. Markov model was modified in order to represent each of the three modes. It was also found that the MIN mode in the passive architecture, where the controller only monitors the two machines but does not take any control actions, is equivalent to the active architecture. Moreover, a case study in [44] showed that in the MIN mode, an increase in the failure rate of the machines has no effect on system unavailability. It also showed that the MIN mode should be avoided from a reliability point of view.

In [45], the Mean Time To Failure (MTTF) of a fault-tolerant two-machine production line is investigated in the context of Networked Control Systems (NCS). Markov model is used and a special metric is introduced in order to increase MTTF by finding the most cost-efficient and practical way of simultaneously decreasing controller failure rate and increasing repair rate and coverage. It was shown that there is more complex approach where the failure rate, repair rate and coverage are not totally independent of each other. It was found that the quality of the controller's software and the machine operators' expertise in the Markov model affect all three parameters mentioned before. Quality of the software installed on the controller is a factor that can affect the failure rate as better version of the software will have a lower software failure rate. A better software version is also expected to have more sophisticated error detection and recovery mechanisms which will increase the coverage. Finally, the diagnostics capabilities of the software should be enhanced which will result in reducing troubleshooting time and decreasing the repair time.

Operator's expertise is another factor which reduces the number of mistakes while operating the machines. Therefore, repair rate will decrease and less time will be required to repair a controller thus increasing the repair rate.

In [46], the effects of failures on the productivity of fault-tolerant networked control systems are investigated under varying loads. Also, Markov models are developed and used to calculate system probabilities which are combined with the maximum speed of operation in each system state. Then the average speed of operation is obtained and Markov models are used to find the best speed mix that would yield maximum output capacity. The average speed of operation at maximum load is compared to that at normal load by using practical numbers for both Mean Time To Failure and the Mean Time To Repair. The case study showed that it is preferable in the fault-free situation to operate the machines at maximum speed and in the case one or more controllers fail situation to operate the machines at normal speed.

In [47], actuator fault-tolerant architecture was presented in order to detect all relevant faults of an electrical steering system by using a double stator AC motor instead of duplicated motors. The paper showed how active control reconfiguration can accommodate all critical faults which were demonstrated on the hardware of a warehouse truck. There are other ways of analyzing the fault-tolerant problem for the networked control systems (NCSs) such as using fuzzy models [48]. The Takagi-Sugeno (T-S) fuzzy model with parametrical uncertainties was used to approximate the T-S model where robust controllers were designed with sensors or actuators failure. It was shown via simulations that the method is effective and the system can be kept asymptotically stable under some sensors failures or actuators failures.

In [49], the 802.11g standard was used without modification in a fault-tolerant networked control system with two or three cascaded work cells. OPNET simulations showed that a two-cell system can tolerate the failure of one of its two controllers even in the presence of noise. For a three-cell system, up to two controllers can fail (in the presence of noise) and the remaining operational controller will be able to handle the load of all three cells. Finally, a performabllity model was developed to simultaneously take into account controller failure data with the risk of not adhering to the required delay constraints. System performability is often used as a tool, where system performance as well as failure data are included within the same metric [50]. Transient (or Point) Performability was used for two and three-cell systems. The first step to calculate TP is the development of a reliability model for the system such as a Continuous Time Markov Chain (CTMC) model. The second step in the calculation of system performability is the assignment of a reward for each state. The reward was equal to the difference between the average delay and the maximum allowable delay.

Finally, The Transient Performability TP(t) is obtained as follows:

TP(t) = $\sum_{i\in\varphi} PiRewi$ where φ is the set of the states in the model and Rewi is the reward of state i. Performability analysis showed that the higher the controller failure rate, the higher the performability.

According to the literature, Ethernet is widely used in Networked Control Systems and proved to be a very successful protocol. One of the common models used in NCS, which is the 16-1-4 machine where there is an in-loop separate controller, was successful in meeting the required time constraints using unmodified Ethernet and running on top of Gigabit Ethernet links . A new approach was presented in the literature where both of the computation and actuation could take place in the same node by integrating both the controller and actuator together. Different fault

tolerant techniques were studied in the literature on the node level (sensors, controllers, and actuators). Finally, perfomability was introduced which can be used in evaluating system performance by adding a reward to each state.

III. Evaluating the Performance of In-Loop vs. S2A Models for Ethernet-Based NCS

III.1 Proposed Model

In this section, a comparison is made between two different control network models. The in-loop model is similar to the one in [7,25] while the second one is the new proposed direct sensor to actuator (S2A) model in this research. In the in-loop model, the controller receives the packets from the sensors and sends control packets to the actuators. The controlling process in the in-loop model takes place in an individual controller node. On the other hand, in the proposed model, the controlling process takes place in the smart actuator node(s) which are more intelligent nodes where both the control and actuation processes occur. Also, there is a supervisor node which is responsible for monitoring network behavior by receiving packets from all the different nodes in the network.

Additionally, the effect of additional load will be studied on the two models. The total end-to-end delay of the in-loop model is expected to be always larger than the proposed model whether operating under Fast or Gigabit operation or even with additional load. This is due to the fact that the traffic sent must go through an additional intermediate hop via the controller thus increasing the delay for the in-loop model. Note that the end-to-end delay includes all types of encapsulation/decapsulation, propagation and queuing delays.

III.2 Models Description

The in-loop model is similar to the one used in [7, 25]; it consists of 16 sensors, one controller and 4 actuators. The controller receives data from the 16

sensors then computes the control action and transmits it to the 4 actuators. Also, the proposed S2A model consists of 16 sensors and 4 actuators but instead of a controller, a supervisor is used. In the S2A model, each one of the sensors sends a packet directly to every actuator via the switch. There is also a supervisor node which receives packets from all the sensors and actuators in the network. The main role of the supervisor node is monitoring the behavior of the network. The main difference between the two models is shown in Figure 4, where one node acts as a controller in the in-loop model while acts as a supervisor in the S2A one.

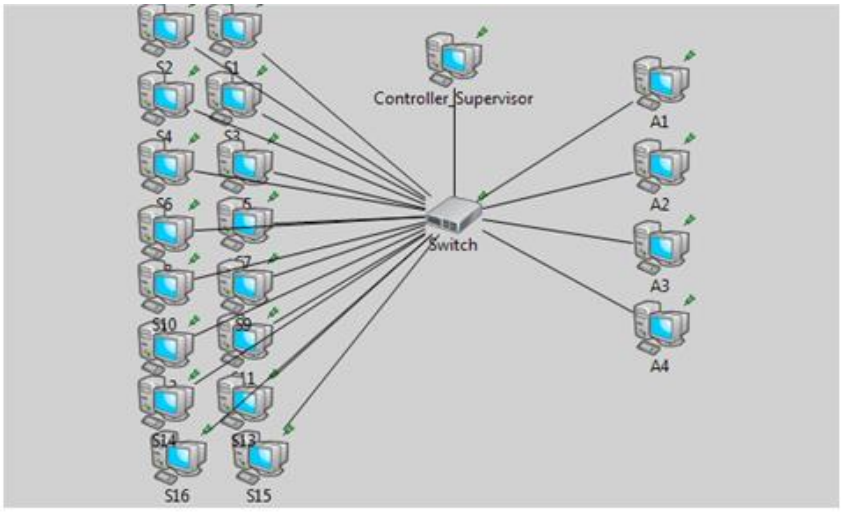

Figure 4: Architecture of both the in-loop model (with controller) and the proposed model (with supervisor)

OMNeT++ [51] is chosen as the simulation platform because it is one of the most widely used network simulators. All the nodes including sensors, controllers, supervisor, and actuators are modeled using standard hosts. Control packets are communicated on-top-of UDP as it is the most suitable for control packets [52]. Also, the payload is fixed at 100Bytes. The sampling frequency used in the two models is 1,440Hz based on a 1440 electric pulses encoder for 360 degrees shaft rotation assuming one revolution per second [53]. Therefore, the control action must be taken within a time frame of 694μs which is the inverse of the sampling

frequency (1440 Hz). Both models are compared once on-top-of Fast Ethernet and again on-top-of Gigabit Ethernet. Additional load is modeled as a TCP application with a flat file size of 500KB between the controller/supervisor and an external node to the networks running all over the simulation time which represents a maintenance engineer communicating with the controller/supervisor. TCP is used for the load because TCP is heavier than UDP due to socket connection, congestion control and reliability.

III.3 ANALYSIS

This subsection presents an analysis to calculate the theoretical total end-to-end delay for both models mentioned above using both Fast and Gigabit switched Ethernet. The presented analysis aims to model, calculate and contrast the end-to-end delays resulting from the periodic nature of the control traffic in both models. A worst-case delay analysis is carried out on both models, therefore the focus will be on the last packet being transmitted by the final sensor node. In other words, all previously sent packets are queued up ahead of the last packet as shown in Figure 5.

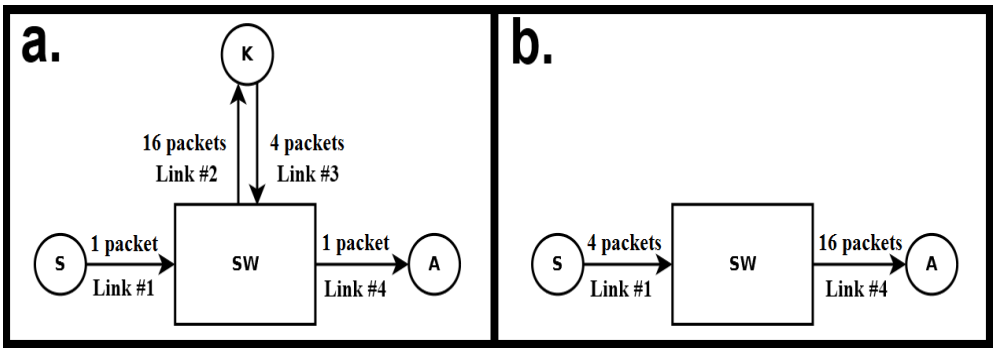

Figure 5: Worst-case packet flow analysis

The total number of packets that must be sent sequentially over each link for each of the two models is calculated using the worst-case packet flow analysis. The number of packets that must be transmitted sequentially for the in-loop model (a. 22

packets) is observed to be larger than the number of packets required by the proposed S2A model (b. 20 packets) as shown in Fig. 5.

For both models, the amount of time required for the transmission of a single packet over a particular link is given by:

$$D_{packet} = D_{transmission} + D_{propagation} + D_{processing} + D_{queuing} \quad (1)$$

According to the literature, the processing delay ($D_{processing}$) is difficult to be calculated, and its value in many cases is legible compared to other types of delays [54].

The queuing delay ($D_{queuing}$) will be reflected through the number of packets calculated in the worst case queuing analysis. The Link Transmission delay ($D_{transmission}$) is the amount of time required for all of the packet's bits to be transmitted onto the link and it is a function of the packet length L (bits) and link transmission rate R (bps) [55].

$$D_{transmission} = L / R \quad (2)$$

The length of the packet is fixed to 100Bytes at the application layer; however, additional packet and frame header overhead approximately 58Bytes ((8)UDP+(20)IP+(30)Ethernet) must be taken into consideration. All the links are Gigabit Ethernet in one scenario and Fast Ethernet in the second scenario, therefore

$$\text{Gigabit Ethernet: } D_{transmission} = (158\times8) / (10^9) = 1.264\mu s \quad (3)$$

$$\text{Fast Ethernet: } D_{transmission} = (158\times8) / (10^8) = 12.64\mu s \quad (4)$$

The propagation delay ($D_{propagation}$) is the time taken for the packet to travel from the sender to the receiver and it is a function of the link length d (m) and the propagation speed s (m/s) [55].

$$D_{propagation} = d / s \quad (5)$$

The length between each node and the switch is d = 1.5m and the transmission speed in the Ethernet links is $s = 2\times10^8$ m/s.

$$D_{propagation} = 1.5 / (2\times10^8) = 0.0075\mu s \quad (6)$$

Finally, The total end-to-end delay for the worst-case packet flow is given by:

$$D_{total} = D_{packet} \times \text{Total Number of Packets Transmitted Sequentially} \quad (7)$$

Therefore, the total end-to-end delay can be calculated by substituting in Equations (1) & (7) as shown below

In- Loop Model

Fast Ethernet: $D_{total} = (22\times (12.64+0.0075) \times10^{-6}) = 278.245\mu s$ (8)

Gigabit Ethernet: $D_{total} = (22\times (1.264+0.0075) \times10^{-6}) = 27.973\mu s$ (9)

S2A Model

Fast Ethernet: $D_{total} = (20\times (12.64+0.0075) \times10^{-6}) = 252.95\mu s$ (10)

Gigabit Ethernet: $D_{total} = (20\times (1.264+0.0075) \times10^{-6}) = 25.43\mu s$ (11)

A summary of the worst-case theoretical results for both in-loop and S2A models is shown in Table 1.

Table 1: Worst-case end-to-end delay analysis results summary (in µs)

Scenario	Link Speed	Theoretical Result	% In-Loop Delay Increase
In-Loop Model	100Mbps	278.245	10 %
	1Gbps	27.973	10 %
S2A Model	100Mbps	252.95	
	1Gbps	25.43	

III.4 SIMULATION RESULTS

III.4.1 WITHOUT ADDITIONAL LOAD

In this section, OMNET simulations are carried out for both models: the in-loop one similar to the model in [25] and the S2A model proposed in this research. The maximum end-to-end delay is found to be 29.245µs for the in-loop model while 26.575µs for the proposed model using Gigabit Ethernet. Note that the 29.245µs delay is the sum of the 22.591µs (maximum sensor to controller end-to-end delay) and 6.655 µs (maximum controller to actuator end-to-end delay) for the in-loop model as shown in Figures 6 and 7. While in the proposed S2A model, the 26.575µs delay is the delay for the actuator node only as shown in Figure 8. This is expected due to the fact that the traffic sent in the in-loop model must go through additional intermediate hops via the controller compared to the S2A model thus increasing the experienced end-to-end delay. Therefore, the proposed S2A model performs better than the in-loop one.

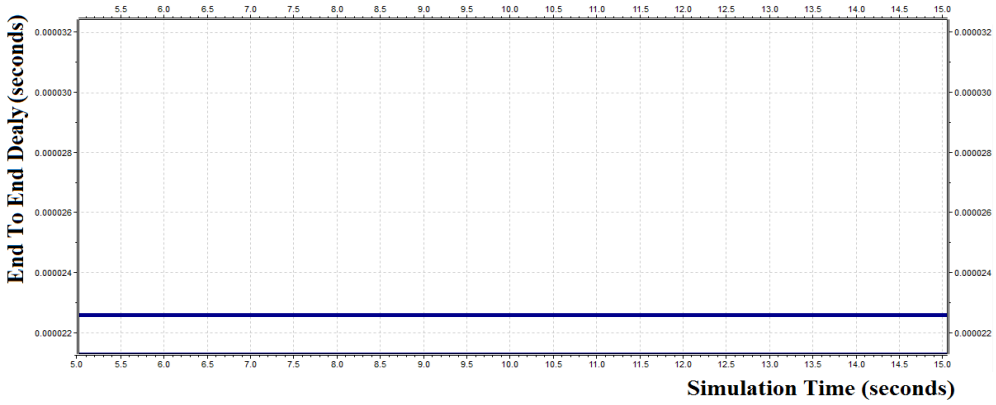

Figure 6: Maximum sensor to controller end-to-end delay using Gigabit Ethernet (In-Loop Model)

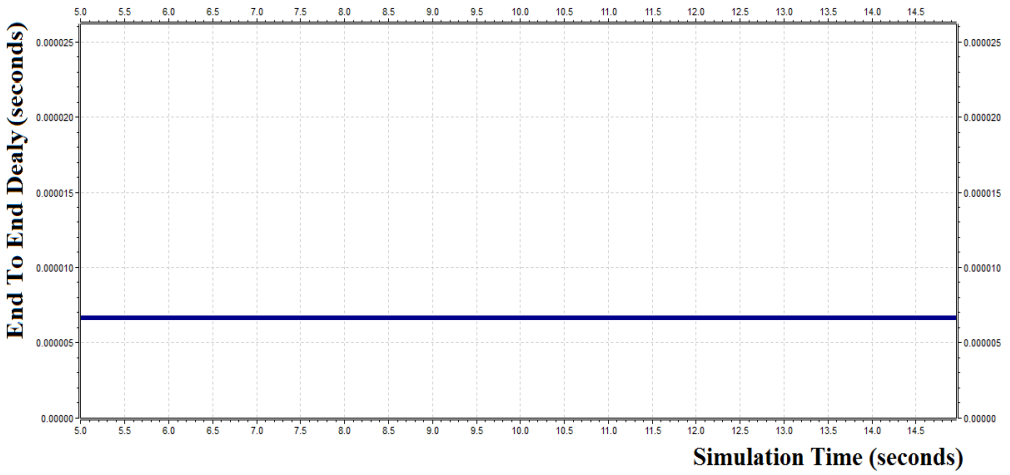

Figure 7: Maximum controller to actuator end-to-end delay using Gigabit Ethernet (In-Loop Model)

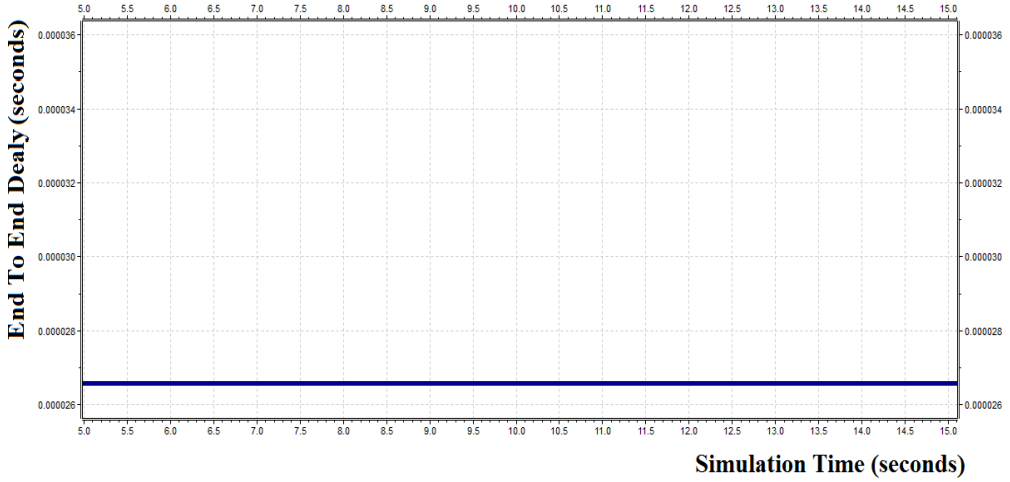

Figure 8: Maximum sensor to actuator end-to-end delay using Gigabit Ethernet (S2A Model)

Similarly, using Fast Ethernet, the proposed S2A model has a smaller maximum end-to-end delay of 265.615µs compared to 292.189µs for the in-loop model. Note that the 292.189µs delay is the sum of the 225.77µs (maximum sensor to controller end-to-end delay) and 66.419 µs (maximum controller to actuator end-to-end delay) for the in-loop model as shown in Figures 9 and 10. While in the proposed S2A model, the 265.615µs delay is the delay for the actuator node only as shown in Figure 11. In the figures, the x-axis represents the Simulation Time (seconds) and the y-axis shows the End-to-end Delay (seconds).

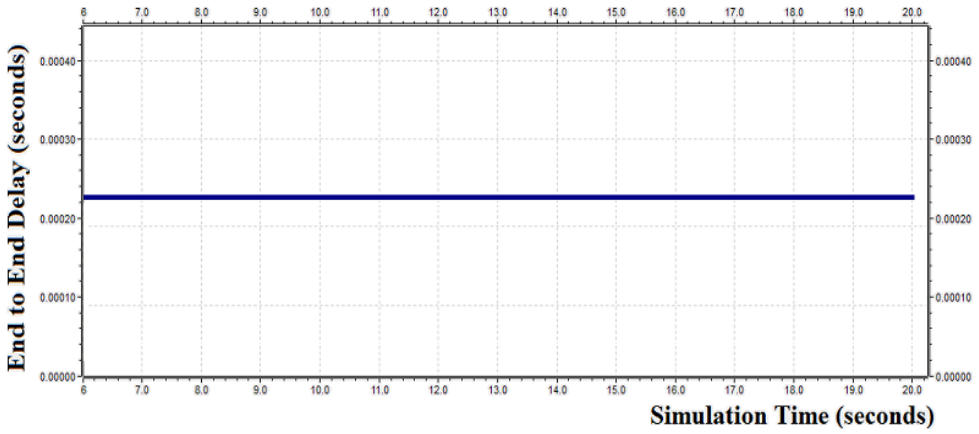

Figure 9: Maximum sensor to controller end-to-end delay using Fast Ethernet (In-Loop Model)

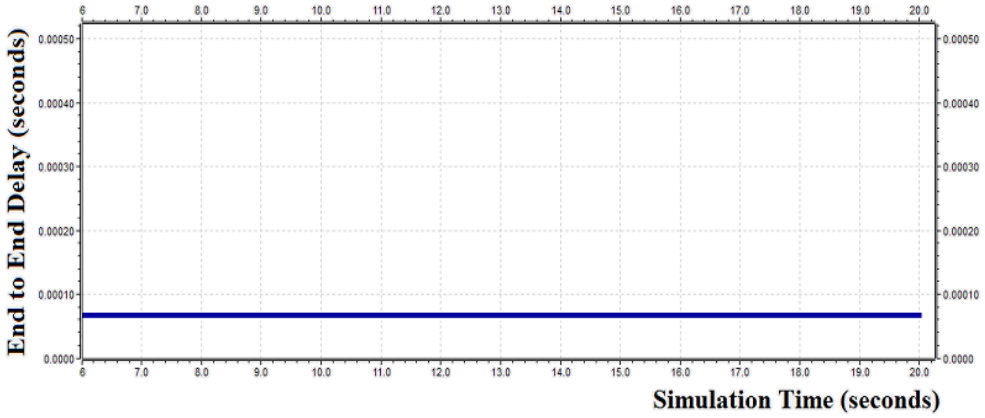

Figure 10: Maximum controller to actuator end-to-end delay using Fast Ethernet (In-Loop Model)

Note that in the in-loop model, the maximum end-to-end delay is the sum of the sensor to controller (Figure 9) and controller to actuator (Figure 10) end-to-end delays. Note that the maximum total end-to-end delay is less than the system's 694µs sampling period and there were no packets dropped.

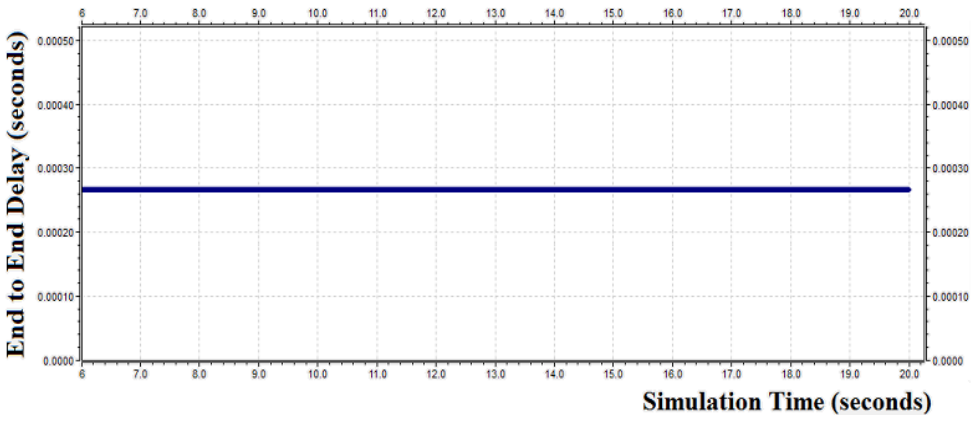

Figure 11: Maximum sensor to actuator end-to-end delay using Fast Ethernet (S2A Model)

The figure shows the constant maximum end-to-end delay in the proposed S2A model between the 4 actuators and the 16 sensor nodes. Note that the observed maximum end-to-end delay is less than the system's 694µs sampling period and there were no packets dropped.

III.4.2 WITH ADDITIONAL LOAD

OMNET simulations are carried out again for both models: the in-loop one and the S2A model. This time an additional TCP load with a flat file size 500KB is added between the controller in the in-loop model or the supervisor in the S2A model and an external node which represents a maintenance engineer communicating with the controller/supervisor. Using Gigabit Ethernet, the maximum end-to-end delay for the in-loop model is increased to 92.44µs. On the other hand in the proposed S2A model, this additional load did not affect the total end-to-end delay. Similarly, in Fast Ethernet, the maximum end-to-end delay for the in-loop model is increased to 976µs (which represents a system failure as it exceeds 694µs which is the delay constraint of the real-time control system). Also, the additional load does not affect the total end-to-end delay of the proposed model as observed before while using Gigabit Ethernet. This is due to the fact that the supervisor node communicates in parallel with the

network and with the external node while the controller node works in series with both the network and the external node thus increasing the delay. In conclusion, the proposed model showed less delay under both Fast and Gigabit with and without additional load as summarized in Tables 2 and 3. Also, Table 2 shows the percentage error between calculated and simulated results for the absence of additional load. Finally, note that the packets experience the same end-to-end delay for each sample due to the regularity of the traffic imposed on the network.

Table 2. Theoretical and Simulation Results without Additional Load (In µSeconds)

Scenario	Link Speed	Theoretical Results	Simulation Results	% Error	% In-Loop Delay Increase
In-Loop Model	100Mbps	278.245	292.189	4.77%	10 %
	1Gbps	27.973	29.245	4.35%	10 %
S2A Model	100Mbps	252.95	265.615	4.77%	
	1Gbps	25.43	26.575	4.31%	

Table 3. Simulation Results with Additional Load (In µSeconds)

Scenario	Link Speed	Simulation Results
In-Loop Model	100Mbps	976
	1Gbps	92.44
S2A Model	100Mbps	265.615
	1Gbps	26.575

IV. FAULT-TOLERANCE

IV.1 FAULT-TOLERANT S2A VS. IN-LOOP CONTROLLER

In this section, two different controller-level fault-tolerant models for Ethernet-based Networked Control Systems are studied using unmodified Fast and Gigabit Ethernet. The first is an in-loop controller model while the second is a direct Sensor to Actuator (S2A) model. A comparison is made between the two different fault-tolerant network models in terms of the total end-to-end delay and packets loss. Comparison is made in the fault-free scenario and the scenario where there are failed controller(s).

IV. 1. 1 PROPOSED MODELS

The in-loop fault-tolerant model is based on the one in [25] while the S2A one is based on [35]. In the in-loop model, the 16 sensors send packets to two controllers where only one of them sends control packets to the actuators while the other one is in hot-standby mode as shown in Figure 12. The controlling process in the in-loop model takes place in an individual controller node, while in the S2A model it takes place in the smart actuator node(s) which are more intelligent nodes where the controller and the actuator are integrated in the same node. In other words, both of the control and actuation processes occur in the same node. Therefore, in order to incorporate fault-tolerance into the S2A model, two controllers will be used per actuator where both controllers receive packets from the sensors but only one of them is chosen, via a multiplexer, to send the control packets to the actuators as shown in Figure 13. Finally, there is a supervisor which is responsible for monitoring network behavior by receiving packets from all the different nodes in the network as shown in Figure 14.

IV. 1.2 MODELS DESCRIPTION

The in-loop fault-tolerant model is based on the one used in [25] as shown in Figure 1. It consists of 16 sensors, two controllers and 4 actuators. Both controllers receive packets from all 16 sensors. One of the controllers is active and the other one is in hot-standby mode. The active controller computes the control action and transmits it to the 4 actuators. Watchdog signals are sent between the two controllers on the network level. If the active controller fails, the hot-standby controller will be alerted via the absence of the watchdog signal; therefore, it would take over and become the active controller.

The S2A model also consists of 16 sensors which send data directly to the 4 actuators but instead of a controller, there is a supervisor. All sensors and actuators send packets to the supervisor node which is responsible for monitoring the behavior of the network. The S2A fault-tolerance model is based on the one used in [35] as shown in Figure 14. Two controllers will be integrated with an actuator in the same node. All 16 sensors send packets to both controllers, and watchdog signals are sent between the two controllers on the circuit level. The two controllers are connected to the actuator (A) via a multiplexer. Note that both of the controllers are integrated in the same node on a circuit board as shown in Figure 14. In the fault-free scenario, if the first controller (K1 in Figure 13) works properly by sending data to the actuator, the second controller becomes inactive by sending '0' to the selection line (S) of the MUX in order not to be chosen. When the first controller fails, the second one will be alerted via the absence of the watchdog signal; therefore it will send a '1' to the selection line of the MUX to be chosen and becomes the active one sending the data to the actuator. Furthermore, the second controller is assumed to be an open circuit output. By using a pull down resistor, '0' will be sent to the MUX selection line to

keep the active controller connected to the output of the MUX and continue receiving the data from the first one as shown in Figure 13.

Note that the watchdog signals are sent in the fault-tolerant S2A model on the circuit level, but they are sent on the network level in the in-loop fault-tolerant model. This is considered an added advantage for the S2A fault-tolerant model as the network would not be congested with the watchdog signals thus decreasing the total end-to-end delays.

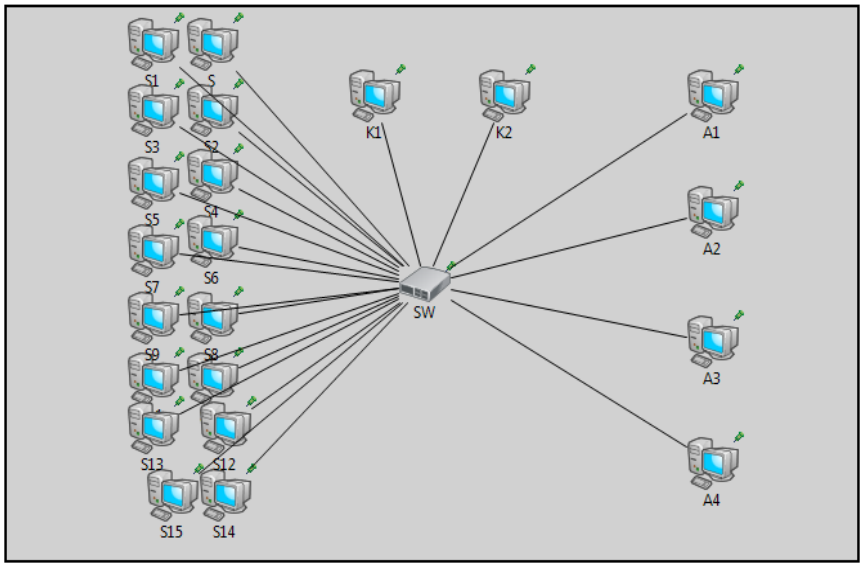

Figure 12: Fault-Tolerant In-Loop Model Architecture.

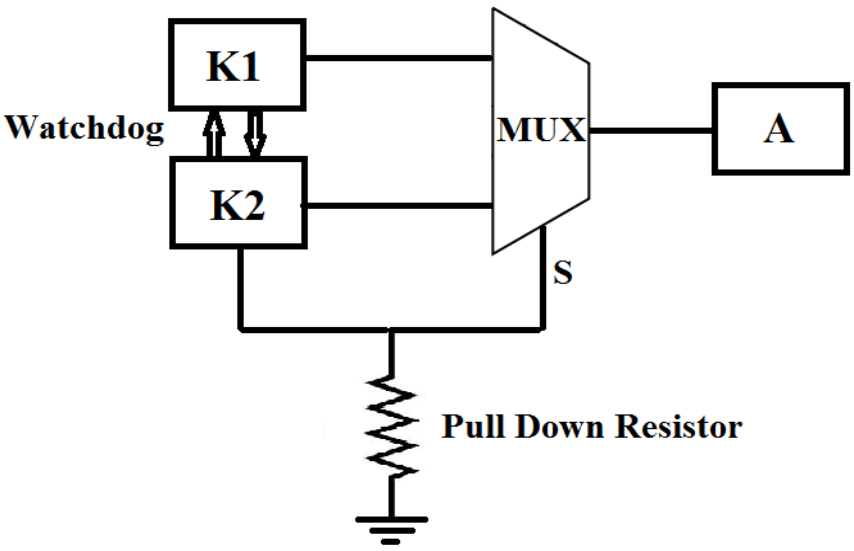

Figure 13: Fault Detection and Recovery Mechanism (S2A Model).

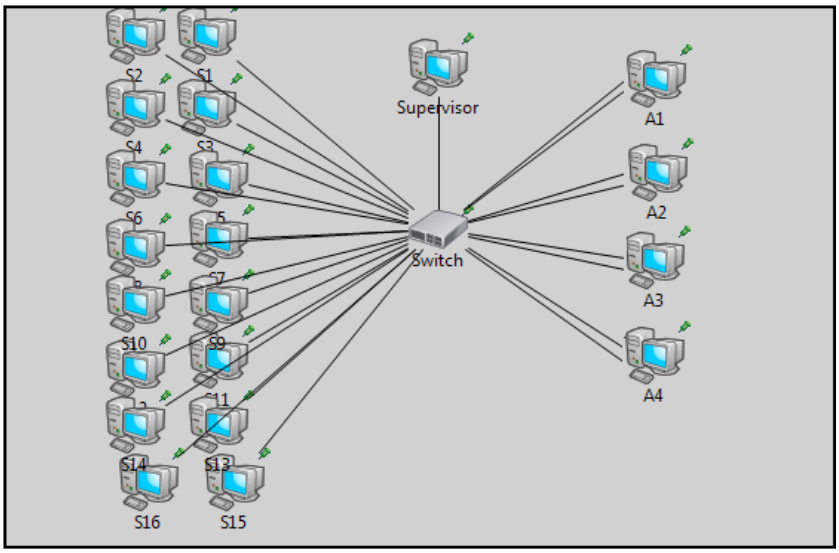

Figure 14: Fault-Tolerant S2A Model Architecture.

OMNeT++ is used as the simulation platform. All the nodes including sensors, controllers, supervisor, and actuators are modeled using standard hosts. Control packets are communicated on-top-of UDP as it is the most suitable for control packets [51]. Also, the payload is fixed at 100Bytes. The sampling frequency used in the two models is 1,440Hz based on a 1440 electric pulses encoder for 360 degrees shaft rotation assuming one revolution per second [52]. Therefore, the control action must be taken within a time frame of 694µs which is the inverse of the sampling frequency (1440 Hz). Watchdog signals are sent over the network in the in-loop model every 347µs which is half of the sampling period in order not to lose any samples when one of the controllers fails. Finally, both models are compared once on-top-of Fast Ethernet and again on-top-of Gigabit Ethernet.

IV.1.3 Analysis

This subsection presents an analysis to calculate the theoretical total end-to-end delay for both models mentioned above using both Fast and Gigabit switched Ethernet. The presented analysis aims to model, calculate and contrast the end-to-end delays resulting from the periodic nature of the control traffic in both models. A

worst-case delay analysis is carried out on both models; therefore the focus will be on the last packet being transmitted by the final sensor node. In other words, all previously sent packets are queued up ahead of the last packet as shown in Figure 15.

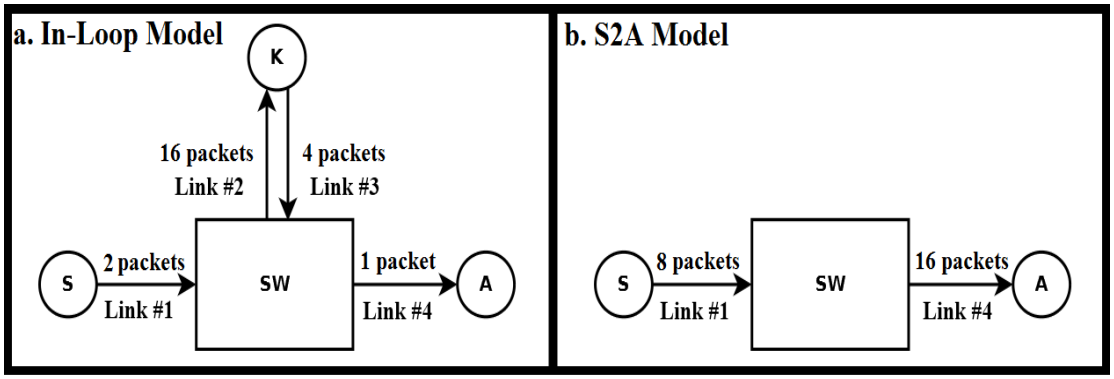

Figure 15: Worst-case packet flow analysis for the fault-free scenario.

In the Fault-Free Scenario, the number of packets is 23 in the in-loop model and 24 in the S2A model as shown in Figure 4, therefore the total delay can be calculated using Equations (1) & (4) & (6) & (7) as shown below

In-Loop Model

Fast Ethernet: D_{total} =

$(23\times (12.64+0.0075) \times 10^{-6}) = 290.893\mu s$ (12)

Gigabit Ethernet: D_{total} =

$(23\times (1.264+0.0075) \times 10^{-6}) = 29.245\ \mu s$ (13)

S2A Model

Fast Ethernet: D_{total} =

$(24\times (12.64+0.0075) \times 10^{-6}) = 303.540\mu s$ (14)

Gigabit Ethernet: D_{total} =

$(24\times (1.264+0.0075) \times 10^{-6}) = 30.516\mu s$ (15)

On the other hand, when one of the controllers fails in the in-loop model and one controller per actuator fails in the S2A model, the worst case packet flow analysis will change to be 22 for the in-loop model and 20 for the S2A model as in [56]. Therefore, the D_{total} calculations can be calculated again as shown below

In-Loop Model

Fast Ethernet: D_{total} =

$(22\times (12.64+0.0075) \times 10^{-6}) = 278.245\mu s$ (16)

Gigabit Ethernet: D_{total} =

$(22\times (1.264+0.0075) \times 10^{-6}) \times 22) = 27.973\mu s$ (17)

S2A Model

Fast Ethernet: D_{total} =

$(20\times (12.64+0.0075) \times 10^{-6}) = 252.95\mu s$ (18)

Gigabit Ethernet: D_{total} =

$(20\times (1.264+0.0075) \times 10^{-6}) = 25.43\mu s$ (19)

A summary of the theoretical results for both in-loop and S2A models in fault-free and failed controller(s) scenarios is shown in Table 4. Fault-Free and Failed Controller(s) scenarios delay increase between the two models is summarized in Table 5.

Table 4: Worst-Case End-to-End Delay Analysis Results Summary (In µs).

Scenario	**Link Speed**	**Fault-Free Theoretical Result**	**Failed Controller(s) Theoretical Result**
In-Loop Model	100Mbps	290.893	278.245
	1Gbps	29.245	27.973
S2A Model	100Mbps	303.540	252.95
	1Gbps	30.516	25.43

Table 5: Fault-Free and Failed Controller(s) scenarios delay increase

Scenario	Link Speed	%Fault-Free S2A Delay Increase	% Failed Controller(s) In-Loop Delay Increase
In-Loop Model	100Mbps		10 %
	1Gbps		10 %
S2A Model	100Mbps	4.34 %	
	1Gbps	4.34 %	

IV. 2 SIMULATION RESULTS

In this section, OMNET++ simulation results are presented. In all simulations, there were no packets dropped.

IV.2.1 FAULT-FREE SCENARIO

OMNeT++ simulations are carried out for both fault-tolerant models: the in-loop based on the model in [25] and the S2A model based on the one in [35]. Using Fast Ethernet, the in-loop model had a smaller maximum end-to-end delay of 305.470µs compared to 318.734µs for the S2A model. Note that the 305.470µs delay is the sum of the 239.055µs (maximum sensor to controller end-to-end delay) and 66.415µs (maximum controller to actuator end-to-end delay) for the in-loop model as shown in Figures 16 and 17 where the data travels over two hops. While the 318.734µs for the S2A model, represents the direct sensor to actuator node delay as shown in Figure 18 where the data travels over one hop. In the Figures, the x-axis represents the Simulation Time (seconds) and the y-axis shows the End-to-end Delay (seconds).

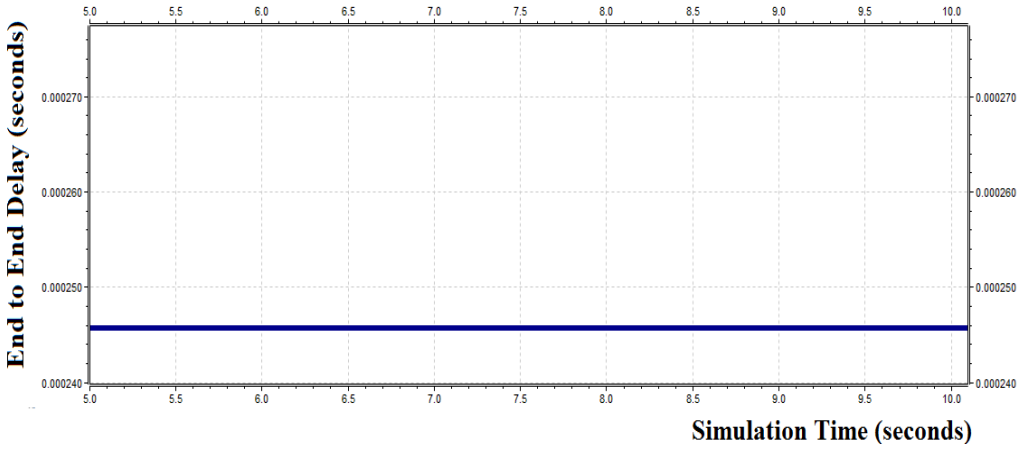

Figure 16: Maximum sensor to controller end-to-end delay using Fast Ethernet (In-Loop Model).

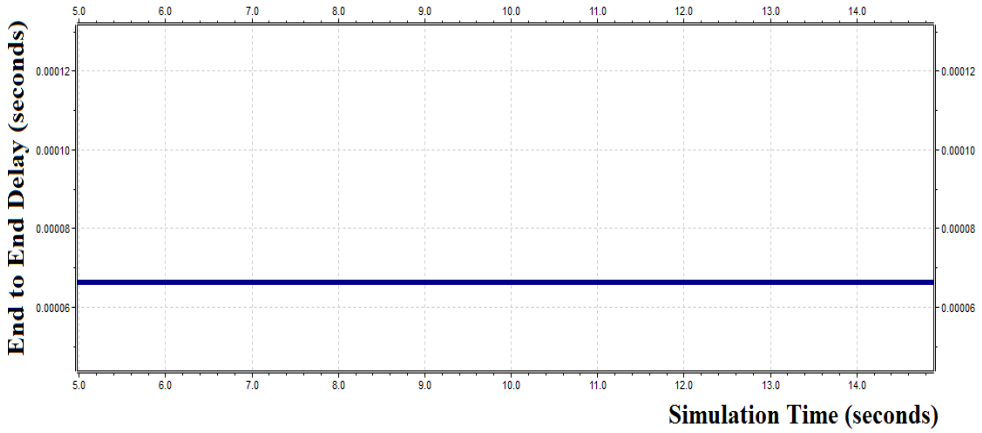

Figure 17: Maximum controller to actuator end-to-end delay using Fast Ethernet (In-Loop Model).

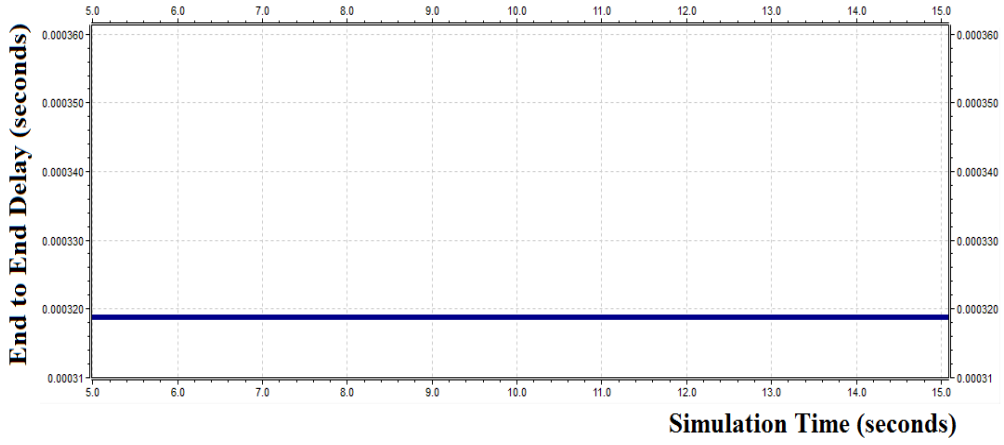

Figure 18: Maximum sensor to actuator end-to-end delay using Fast Ethernet (S2A Model).

Similarly, using Gigabit Ethernet, the maximum end-to-end delay was found to be 30.574µs for the in-loop model and 31.886µs for the proposed S2A model. Note that the 30.574µs delay is the sum of the 23.919µs (maximum sensor to controller

end-to-end delay) and 6.655µs (maximum controller to actuator end-to-end delay) for the in-loop model as shown in Figures 19 and 20 where the data travels over two hops. While the 31.886µs for the S2A model, represents the direct sensor to actuator node delay as shown in Figure 21 where the data travels over one hop. This means that the in-loop model performs better, which is expected, due to the fact that there are two separate controllers in the in-loop model while there are 8 controllers (2 controllers per actuator) in the S2A thus increasing the amount of traffic in the network which increases the experienced end-to-end delay.

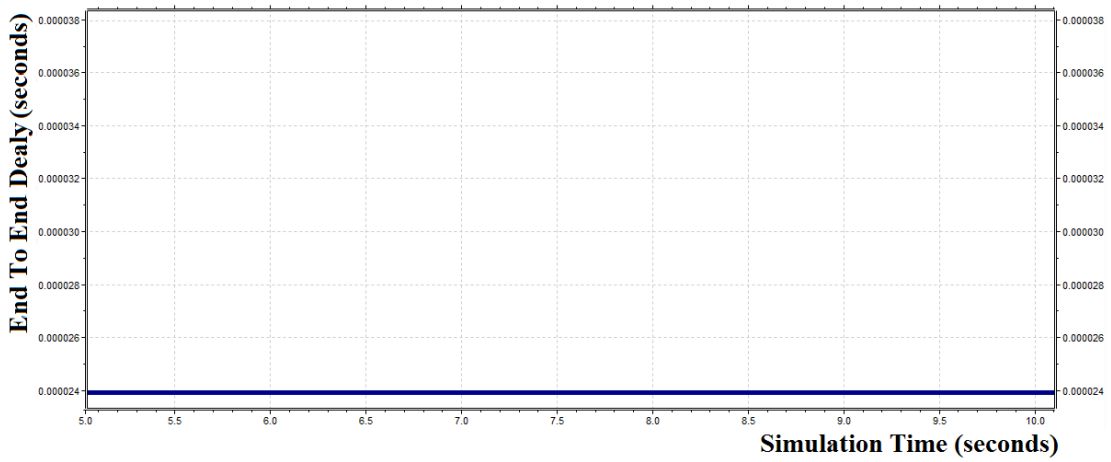

Figure 19: Maximum sensor to controller end-to-end delay using Gigabit Ethernet (In-Loop Model).

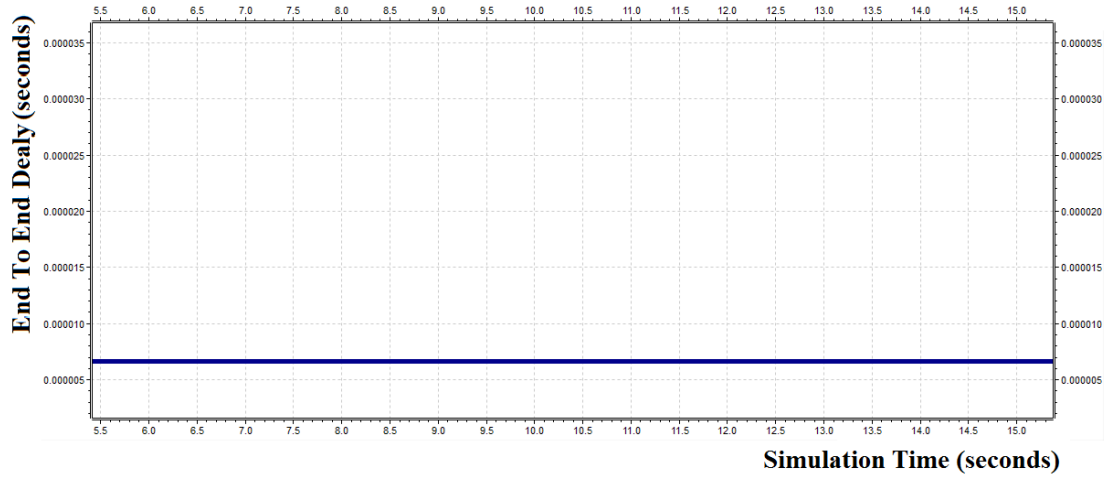

Figure 20: Maximum controller to actuator end-to-end delay using Gigabit Ethernet (In-Loop Model).

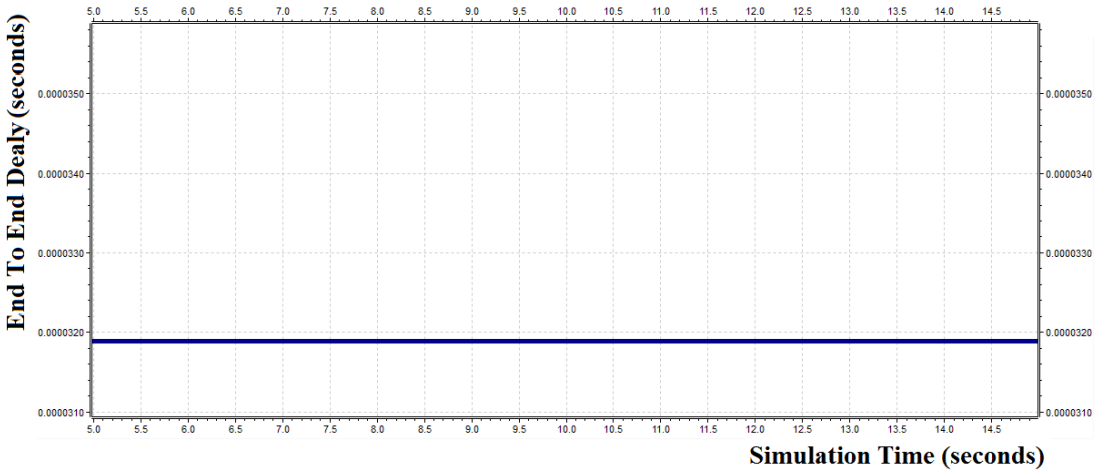

Figure 21: Maximum sensor to actuator end-to-end delay using Gigabit Ethernet (S2A Model).

IV.2.2 SCENARIO WITH THE FAILED CONTROLLER(S)

On the other hand, if one of the two separate controllers in the in-loop model fails or one controller from each of the four pairs of integrated controllers in the S2A model fails, it was found that the S2A model performs better with less end-to-end delay. This is due to the fact that traffic sent in the in-loop model must go through additional intermediate hops via the controller. While in the S2A model, only one hop is needed to transmit the traffic thus decreasing the experienced end-to-end delay. In the scenario with the failed controller(s), using Fast Ethernet, the S2A model had a smaller maximum end-to-end delay of 265.615µs compared to 292.189µs for the in-loop model in Figure 24. Note that the 292.189µs delay is the sum of the 225.77µs (maximum sensor to controller end-to-end delay) in Figure 22 and 66.419 µs (maximum controller to actuator end-to-end delay) in Figure 23 for the in-loop model.

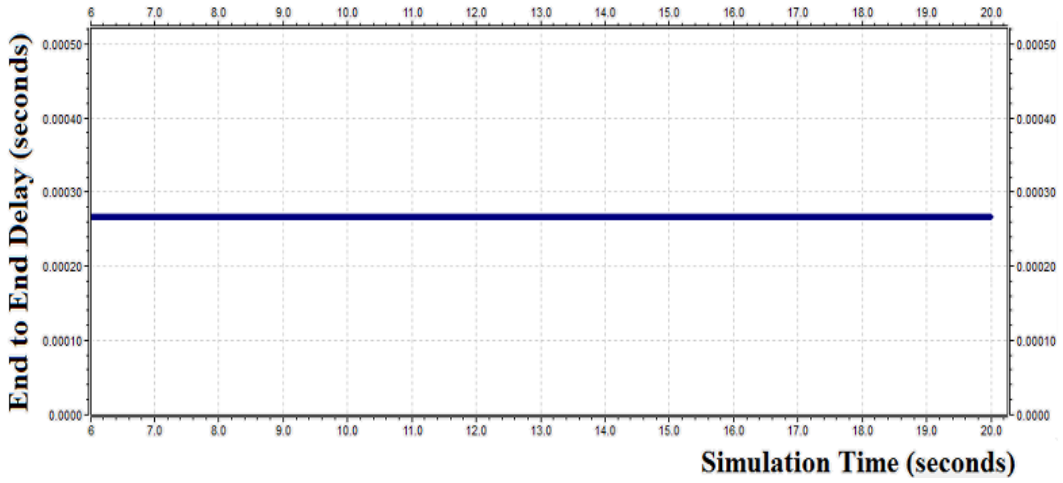

Figure 22: Maximum sensor to controller end-to-end delay using Fast Ethernet (In-Loop Model)

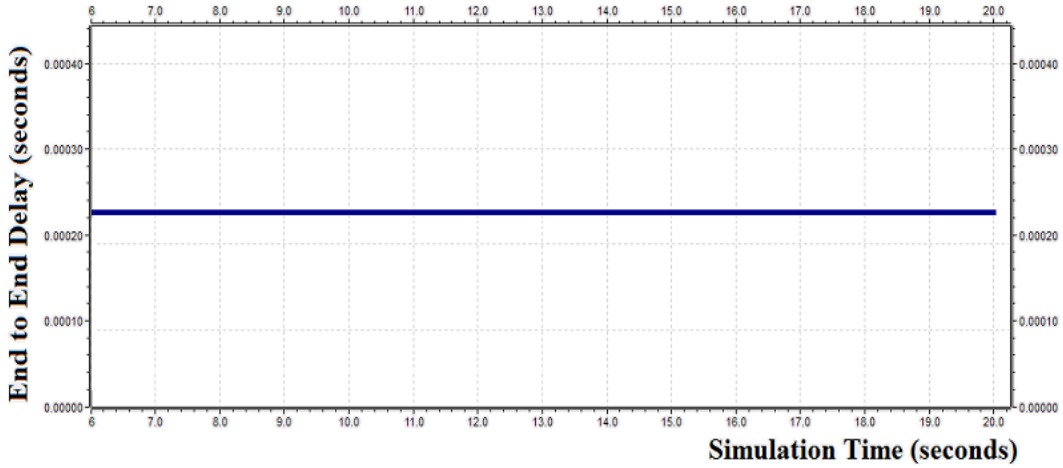

Figure 23: Maximum controller to actuator end-to-end delay using Fast Ethernet (In-Loop Model)

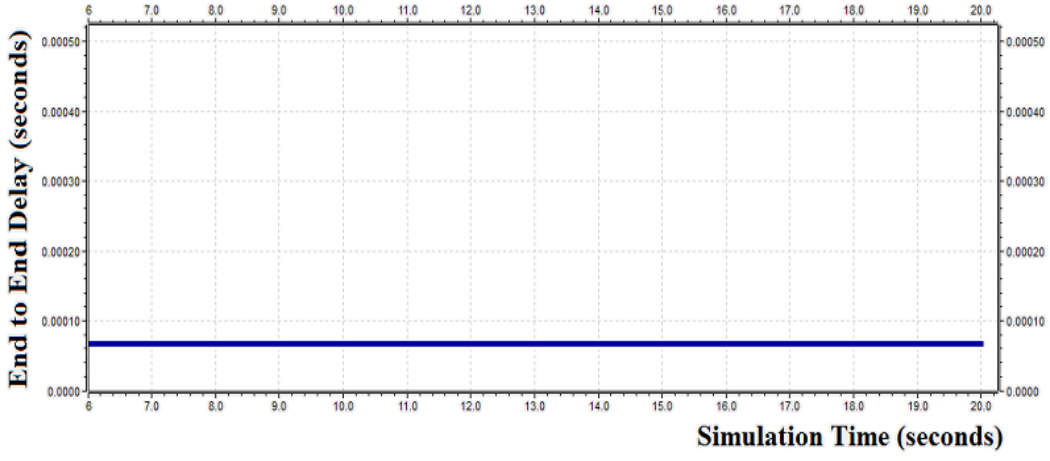

Figure 24: Maximum sensor to actuator end-to-end delay using Fast Ethernet (S2A Model)

Similarly, using Gigabit Ethernet, the maximum end-to-end delay was found to be 29.245µs for the in-loop model and 26.575µs for the S2A model as shown in figure 25. Note that the 29.245µs delay is the sum of the 22.591µs (maximum sensor to controller end-to-end delay) in Figure 26 and 6.655 µs (maximum controller to actuator end-to-end delay) in Figure 27 for the in-loop model.

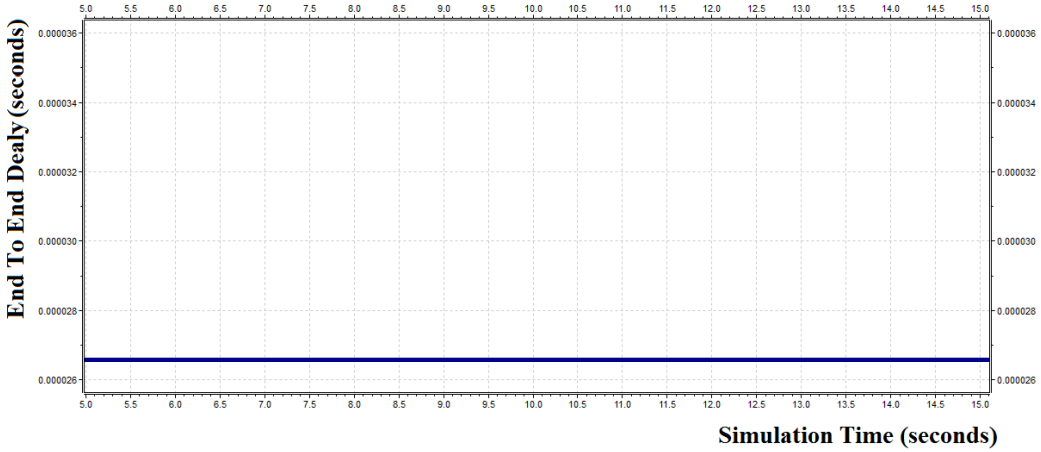

Figure 25: Maximum sensor to actuator end-to-end delay using Gigabit Ethernet (S2A Model)

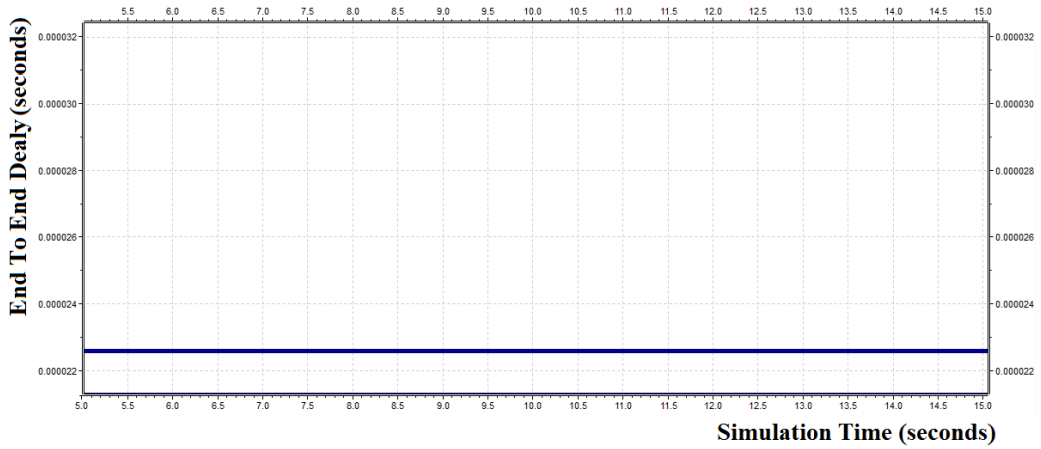

Figure 26: Maximum sensor to controller end-to-end delay using Gigabit Ethernet (In-Loop Model)

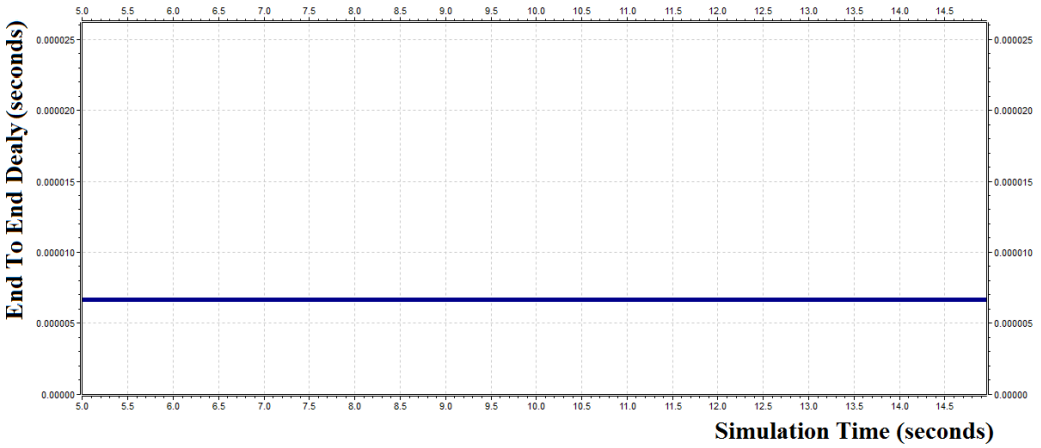

Figure 27: Maximum controller to actuator end-to-end delay using Gigabit Ethernet (In-Loop Model)

In conclusion, using both Fast and Gigabit Ethernet, the in-loop model showed less delay in the fault-free scenario while the S2A model showed less delay in the scenario with the failed controller(s) as summarized in Tables 6 and 7. Also, Tables 6 and 7 show the percentage error between calculated and simulated results. Note that, due to the regularity of the traffic imposed on the network, packets experience the same end-to-end delay for each sample.

Table 6: Theoretical and Simulation results in Fault-Free Scenario (In µs).

Scenario	Link Speed	Theoretical Result	Simulation Result	Error %
In-Loop Model	100Mbps	290.893	305.470	4.77%
	1Gbps	29.245	30.574	4.34%
S2A Model	100Mbps	303.540	318.734	4.76%
	1Gbps	30.516	31.886	4.29%

Table 7: Theoretical and Simulation results in Scenario with the failed controller(s) (In µs).

Scenario	Link Speed	Theoretical Result	Simulation Result	Error %
In-Loop Model	100Mbps	278.245	292.189	4.77%
	1Gbps	27.973	29.245	4.35%
S2A Model	100Mbps	252.95	265.615	4.77%
	1Gbps	25.43	26.575	4.31%

V. PERFORMABILITY ANALYSIS

Fault-tolerance is a hot-topic in many research fields, due to the advantages of a fault-tolerant system over a normal system. Fault-tolerant system is one that can 'tolerate' a fault in one or more components. The system can continue operation, maybe with degraded performance, but will not fail. Down-time can be extremely costly; therefore a system which can tolerate a failure of one or more components while maintaining operation is extremely appealing. The advantage of fault-tolerance of any form in an industrial application is reducing downtime. There are techniques to quantify the increased reliability of the system, such reliability modeling. Another metric that can be analyzed is performability with its various forms: Steady State, Transient and cumulative performability (SSP, TP and CP respectively) and typically relates failure-rates to rewards at different system states.

A comparison is made between two different control network fault-tolerance models. The in-loop fault-tolerant model is based on the one in [25] while the S2A one is based on [56]. In the in-loop model, there are two controllers which receive the packets from the sensors and only one of them sends control packets to the actuators while the other one is in hot-standby mode. In the S2A model, a supervisor is responsible for monitoring network behavior by receiving packets from all the different nodes in the network. The controlling process in the in-loop model takes place in an individual controller node, while in the S2A model it takes place in the smart actuator node(s) which are more intelligent nodes where both the control and actuation processes occur. To incorporate fault-tolerance into the S2A model, two controllers will be used per actuator where both controllers receive packets from the sensors but only one of them is chosen, via a multiplexer, to send the control packets to the actuators.

Reliability and Performability analysis between the two models will be studied and compared using fast Ethernet links relating failure data with reward, depending on the system state. In Networked Control Systems, meeting control system's deadline is essential and failing to meet the deadline is considered system failure. Therefore, the reward is considered to be how far is the total end-to-end delay in each state in the model from the deadline which is 694 μs. The effect of parameters such as failure rate (λ), repair rate (μ) and coverage (c) on performability will be studied. Note that failure rate $(\lambda) = \frac{1}{\text{MTTF}}$ where MTTF is Mean Time To Failure and repair rate $(\mu) = \frac{1}{\text{MTTR}}$ where MTTR is Mean Time To Repair. The probability of successful detection/reconfiguration is called *coverage* [57-59]. The *coverage* (c) is defined as the proportion of faults from which a system can automatically recover [60]. The coverage is included in reliability/availability models and it is determined by the user. Any small mistake in the calculation of the coverage leads to false reliability/availability estimations [59]. It is expected that, if the coverage of a system decreases, system reliability is expected to decrease as well.

The Markov model that represents the case of in-loop model is shown in Figure 28. There are two controllers which receive the packets from the sensors and only one of them sends control packets to the actuators while the other one is in hot-standby mode.

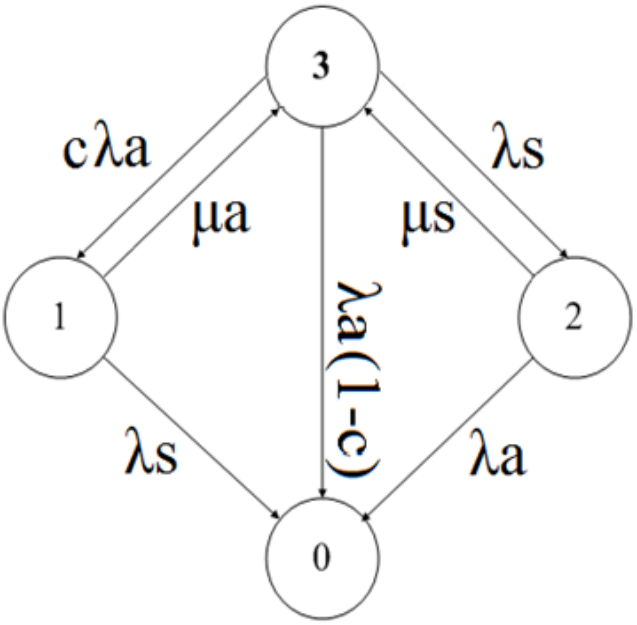

Figure 28:Markov Model of the in-loop model.

There are four states which describe the model:

0: when both controllers fail

1: when the active controller (which receive and send packets) fails

2: when the hot stand-by controller fails

3: when both controllers work properly

where the following rates and parameters are used:

λa, λs: failure rates of the active and stand-by controllers respectively

μa, μs: repair rates of the active and stand-by controllers respectively

c: coverage

Let Ps(t) be the probability of residing in state s (s =3, 2, 1, 0) at time t. The transient probability of residing in any of the four states can be calculated using the following Chapman-Kolmogorov equations [61]:

$$\frac{dp}{dt} = P \times T$$

[P3' P2' P1' P0'] = [P3 P2 P1 P0] × T

where P is the probability transition matrix with length 4 (number of states) while T is the rate transition matrix as show below where each element (i,j) represents the transition rate from i to j where (i,j=0,1,2,3). Note that the sum of the rates per row = 0, therefore the diagonal where (i=j) is (1 - (sum of rates per row)).

$$T = \begin{bmatrix} -(\lambda s + \lambda a) & \lambda ac & \lambda s & \lambda a(1-c) \\ \mu a & -(\lambda s + \mu a) & 0 & \lambda s \\ \mu s & 0 & -(\lambda a + \mu s) & \lambda a \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The probabilities in initial condition is [P3 P2 P1 P0] = [1 0 0 0]

Therefore the kolmogorov differential equations will be

$$\frac{dp3}{dt} = -(\lambda s + \lambda a) \times P_3(t) + \mu a \times P_2(t) + \mu s \times P_1(t) \quad (20)$$

$$\frac{dp2}{dt} = (\lambda ac) \times P_3(t) - (\lambda s + \mu a) \times P_2(t) \quad (21)$$

$$\frac{dp1}{dt} = (\lambda s) \times P_3(t) - (\lambda a + \mu s) \times P_1(t) \quad (22)$$

$$\frac{dp0}{dt} = \lambda a(1-c) \times P_3(t) + \lambda s \times P_2(t) + \lambda a \times P_1(t) \quad (23)$$

Then Matlab is used to solve the differential equations by getting laplace inverse then finding the probability of each state

Then The Transient Performability TP(t) is obtained as follows:

$TP(t) = \sum_{s \in x} PsRews$

Where x is the set of the four states in the model and Rew(s) is the reward of state s.

As said before, the reward is considered how far the total end-to-end delay of each state (sum of sensor to controller and controller to actuator delays) from the deadline which is 694 µs and the reward numbers are summarized in the Table 8 by subtracting the total end-to-end delay from the 694 (Deadline):

Table 8: Reward of each state (In-Loop Model).

State	Total End-to-End Delay	Reward (µs) (694 - Delay)
3	305.47	388.53
2	292.189	401.811
1	292.189	401.811

Note that at state 3, the Total End-to-End delay taken is the maximum between the active and hot stand-by controllers while in states 1 & 2 the delay is equal because in both cases only one controller works. Performability is calculated using Matlab as shown below

```
syms t s la ls ma ms c
A = [(s+ls+la), (-c.*la), (-ls), (la.*(c-1)); (-ma), (s+ls+ma), 0, (-ls); (-ms), 0, (s+ms+la), (-la); 0, 0, 0, s]
g = [1, 0, 0, 0 ] * inv(A)
g=simple(g)
g=vpa(g,10); % ten digits precision
f = ilaplace(g)
f(1) = f(1) .* 388.53;
f(2) = f(2) .* 401.811;
f(3) = f(3) .* 401.811;
f(4) = f(4) .* 0;
performability = sum(f);
```

In the S2A Model, it would be easier to calculate the reliability using combinatorial models compared to Markov Models. In combinatorial models, module failures are independent, failed module produce incorrect results and cannot return to a functional state unless it is repaired. In the S2A Model, two controllers will be used per actuator where both controllers receive packets from the sensors but only one of them is chosen, via a multiplexer, to send the control packets to the actuators. Both controllers are integrated in the same node on a circuit board; therefore upon the failure of one of the controllers, it cannot be repaired again. There are 4 actuators in the S2A Model, therefore there are 8 controllers in the system. The system can be seen as a mixed combinatorial system where each pair of controllers per actuator work in parallel and the four actuators work in series with each other as seen in Figure 29 which represents a reliability block diagram for the S2A fault-tolerant system understudy [61].

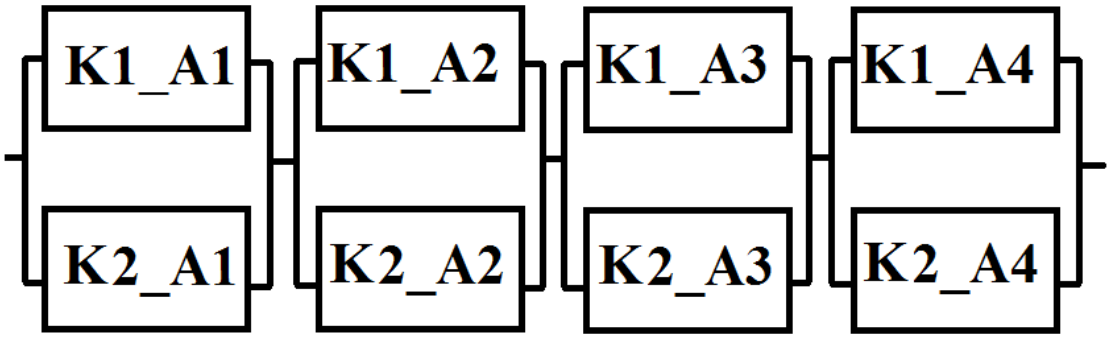

Figure 29: S2A Model Combinatorial System.

Failure only takes place when both controllers integrated in the same actuator fail, therefore, each pair of controllers is connected in series with the other pairs. While every two controllers per actuator are connected in parallel, therefore the minimum number of controllers for the system to still operate is 4 where only one controller failed in each pair of controllers. In order to calculate the total reliability of

the S2A Model R_{system}, the following equation will be used assuming all controllers are identical and independent form each other

$$R_{system} = (R_{A1/A2/A3/A4})^4 = (R_1 + cR_2\,(1\text{-}R_1))^4 \quad (24)$$

where c is the coverage

The first term of the equation (R_1) is the probability that the first controller survives. While the second term (1- R_1) is the probability that the first controller fails, but the second controller (R_2) is still functioning and a successful switchover was accomplished and is donated by the coverage c [61].

Assuming all the controllers have the same failure rate λ , they would have the same reliability R (R1 = R2)

Therefore, the system reliability equation can be rewritten to be

$$R_{system} = (R_{A1/A2/A3/A4})^4 = (R + cR\,(1\text{-}R))^4$$

In order to calculate the performability of the S2A Model, the probability of each state is calculated first. Assuming one actuator, the probabilities will be calculated as shown below in Table 9.

Table 9: Probability of each state (S2A Model).

State	Probability
11	R^2
10	R(1-R)
01	cR(1-R)
00	(1-R)(1-cR)

Therefore, $R_{system} = P(11) + P(10) + P(01) = 1 - P(00) = 1 - (1\text{-}R\text{-}cR\text{+}cR^2) = R\text{+}cR\text{-}cR^2$ = R+cR(1-R) which is equivalent to R_{system} in Equation 24

But in the S2A Model, there are 8 controllers, therefore there are 2^8 = 256 probabilities where many will be equivalent to each other:

For example p (11011000) = p (11) * p (01) * p (10) * p (00)

= $R^2 \times cR(1-R) \times R(1-R) \times (1-R)(1-CR)$

The Transient Performability TP(t) is obtained as follows:

TP(t) = $\sum_{s \in x} PsRews$

Where x is the set of the five states in the model and Rew(s) is the reward of state s where s (number of controllers working) = 4, 5, 6, 7, 8. The minimum number of controllers is 4 (one operational in each pair) because upon the failure of the fifth controller the system would be failed.

As said before, the reward is considered to be how far is the total end-to-end delay of each state (direct sensor to actuator delay) from the deadline which is 694 µs and the reward values are summarized in the following Table 10 by subtracting the total end-to-end delay from the 694 (Deadline):

Table 10: Reward of each state (S2A Model).

Number of controllers working	Total End-to-End Delay	Reward (µs) (694 - Delay)
4	265.614978	428.385022
5	278.894977	415.105023
6	292.174976	401.825024
7	305.454957	388.545043
8	318.734974	375.265026

Performability is calculated using Matlab as shown below:

```
pairs = 4;
vector = zeros(2^(2*pairs), 1)
alive = zeros(2^(2*pairs), 1)
kresult = sym(ones(2^(2*pairs), 1))
for i = 1:length(kresult)
   alive(i) = 0;
   for p = 1:pairs
      j = mod(floor((i-1)/(4^(p-1))), 4)
      if j == 0 %00 - Both Failed
         kresult(i) = kresult(i) .* (1-R)*(1-C*R) .* 0
      elseif j == 1 %01 - Secondary Alive
         kresult(i) = kresult(i) .* C*R*(1-R)
         alive(i) = alive(i) + 1;
      elseif j == 2 %10 - Primary Alive
         kresult(i) = kresult(i) .* R*(1-R)
         alive(i) = alive(i) + 1;
      elseif j == 3 %11 - Both Alive
         kresult(i) = kresult(i) .* R^2;
         alive(i) = alive(i) + 2;
      end
  end
 %   Reward Calculation
 %   Based on number of actuators alive
 %   Substitute rewards (simulation results) here !
```

```
    if alive(i) == 4
        kresult(i) = kresult(i) .* 428.385022;
    elseif alive(i) == 5
        kresult(i) = kresult(i) .* 415.105023;
    elseif alive(i) == 6
        kresult(i) = kresult(i) .* 401.825024;
    elseif alive(i) == 7
        kresult(i) = kresult(i) .* 388.545043;
    elseif alive(i) == 8
        kresult(i) = kresult(i) .* 375.265026;
    else
        kresult(i) = kresult(i) .* 0;
    end
    vector(i) = i;
end
performability = simple(sum(kresult))
```

Comparison will be made between the performability of the in-loop model and the S2A model where the effects of the failure rate (λ), the repair rate (μ) and the coverage (c) will be studied by trying different practical numbers from the industry. In order to compare between the two models, the failure rate (λ) used in the in-loop Model = 4 $\times$ failure rate (λ) used in the S2A Model because in the in-loop model there is one fault-tolerant controller compared to 4 fault-tolerant controllers in the 4 actuators. For example, if $\lambda_{\text{in-loop}}$ = 1/Month, λ_{S2A} = .25/Month.

The effect of the failure rate λ will be studied first and its effect on performability for both models as shown in the following case studies graphs.

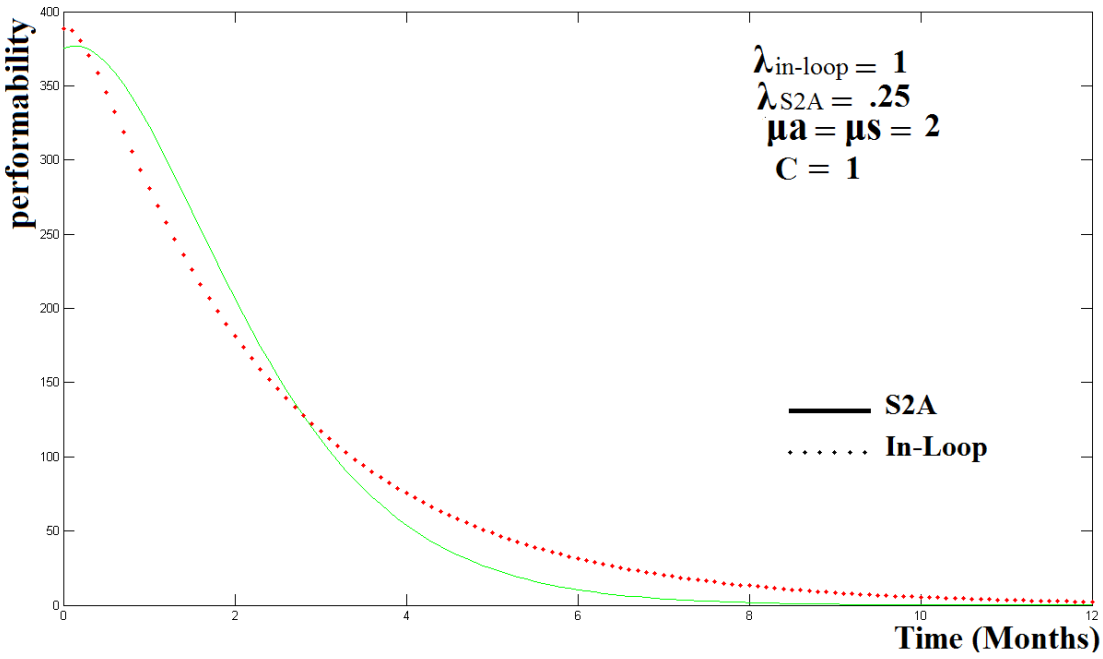

Figure 30: Performability S2A vs. In-Loop Model case study 1

It can be seen in Figure 30 that at $\lambda_{\text{in-loop}}$ = 1/Month, λ_{S2A} = .25/Month, μa = μs = 2/month, c = 1, the S2A model has better performability than the in-loop model in the first 3 months then after t > 3 months the in-loop model starts to perform better. Note that at the beginning at t = 0, the in-loop model seems to perform better than the S2A model because the reward of 2 controllers working = 388.53 which is greater than the reward of the 8 controllers working in the S2A Model which is = 375.265026. Then, the failure rate $\lambda_{\text{in-loop}}$ is increased to 2/Month, while λ_{S2A} = .5/Month, μa = μs = 2/month, c = 1, then performability is calculated for the two models.

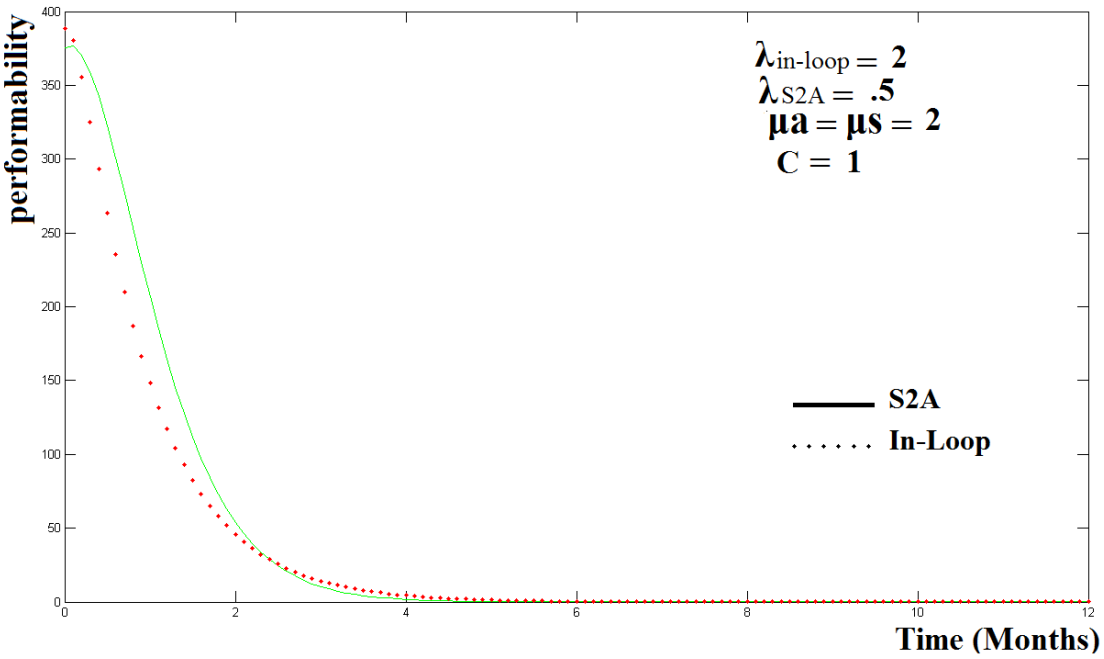

Figure 31: Performability S2A vs. In-Loop Model case study 2

Figure 31 shows that the performability of the S2A Model increases while the performability of the in-loop model decreases with the increase of failure rate λ. The same trend is observed when λ increases to 4 / month as shown in Figure 32. It can also be seen that the cut off point (where both models intersect) occurs earlier with the increase of failure rate. Therefore, the S2A model has better performablity for short mission periods than the in-loop model which means that if you would not like to stop your machine for maintenance or repair then the in-loop would be the one to use. While if you can afford stopping the machine for maintenance then the S2A would give a better performability.

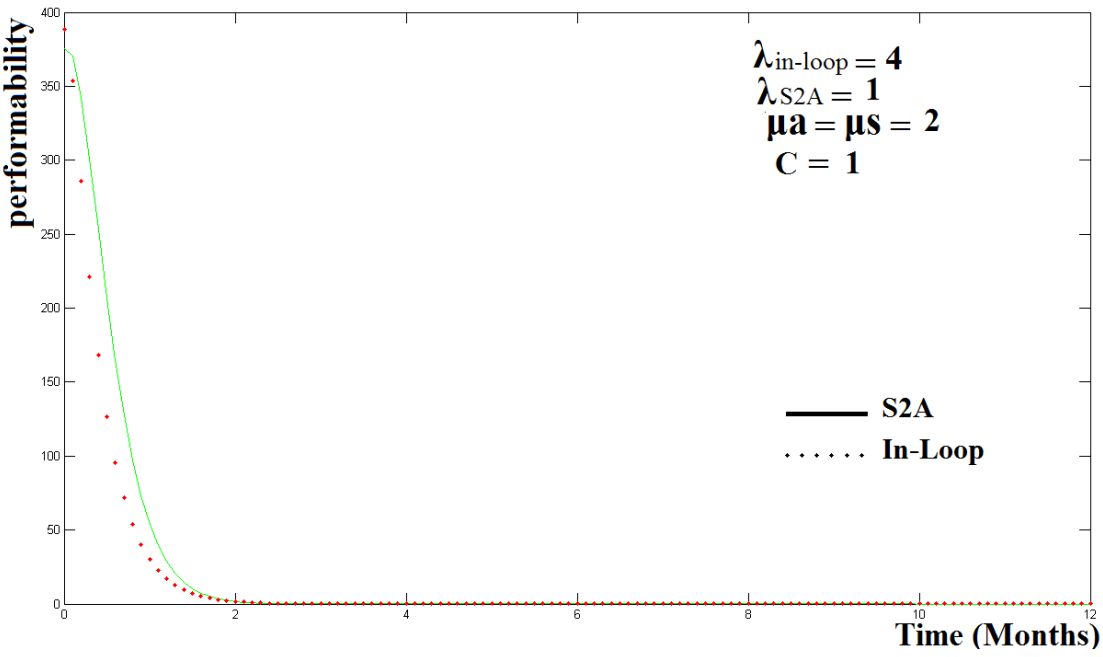

Figure 32: Performability S2A vs. In-Loop Model case study 3

Now, the effect of the Repair Rate μ will be studied and its effect on performability for both models as shown in the following graphs.

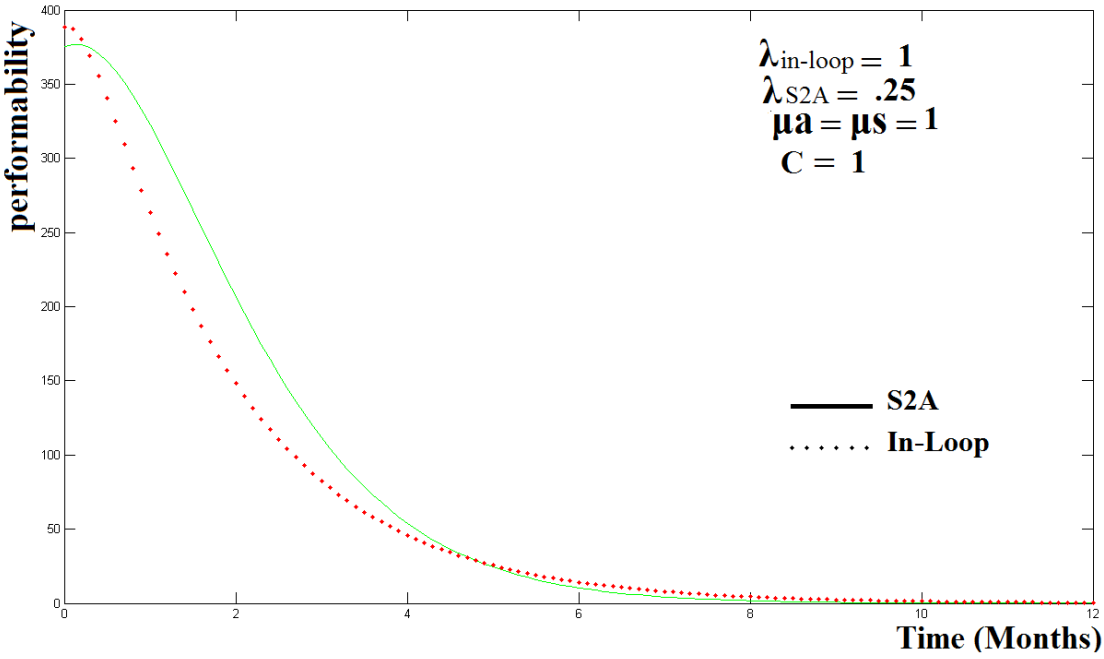

Figure 33: Performability S2A vs. In-Loop Model case study 4

It can be seen in Figure 33 at $\lambda_{in\text{-}loop}$ = 1/Month, λ_{S2A} = .25/Month, μa = μs = 1/month, c = 1, the S2A model has better performability than the in-loop model in the first 4.5 months then after t > 4.5 the in-loop model starts to perform better. Then, the repair rate μ is increased to 2/Month, while $\lambda_{in\text{-}loop}$ = 1/Month, λ_{S2A} = .25/Month, c = 1, then performability is calculated for the two models as shown in Figure 34

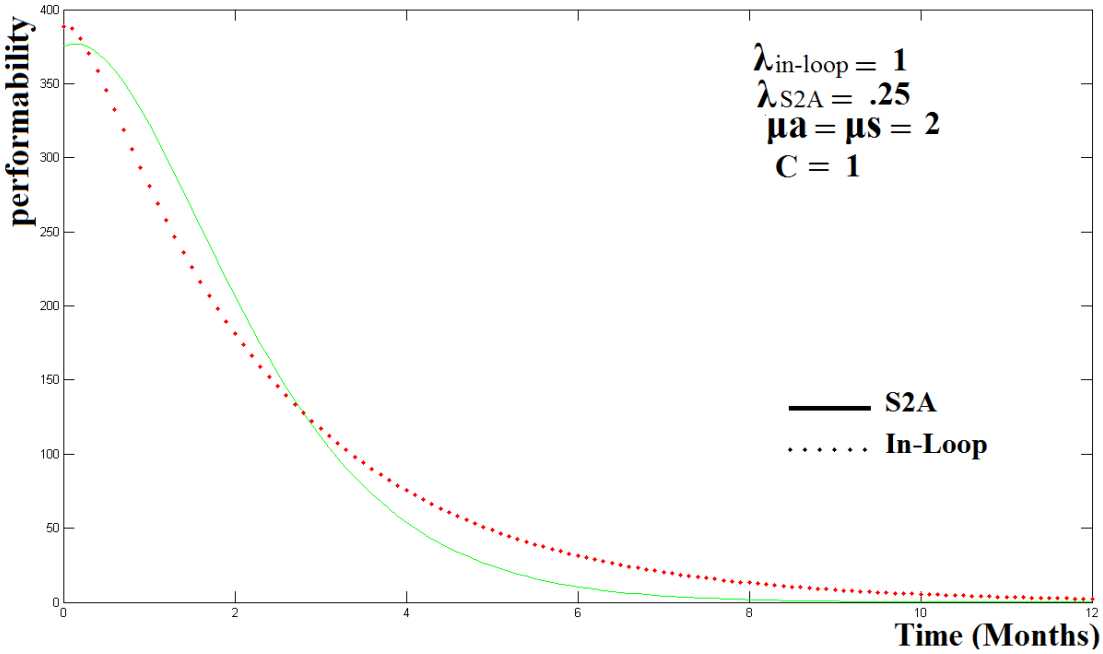

Figure 34: Performability S2A vs. In-Loop Model case study 5

It is found that the performability of the S2A Model does not change while the performability of the in-loop model increases with the increase of the repair rate μ. The same trend is observed when λ increases to 4 as shown in Figure 35. It can be see that the repair rate affects the performablity of the in-loop model which is expected compared to the S2A model where there is no repair as it would be hard to repair the failure of any of the two controllers as they are integrated together in the same board. Therefore, it would be better to use the in-loop model if the machine can be repaired frequently.

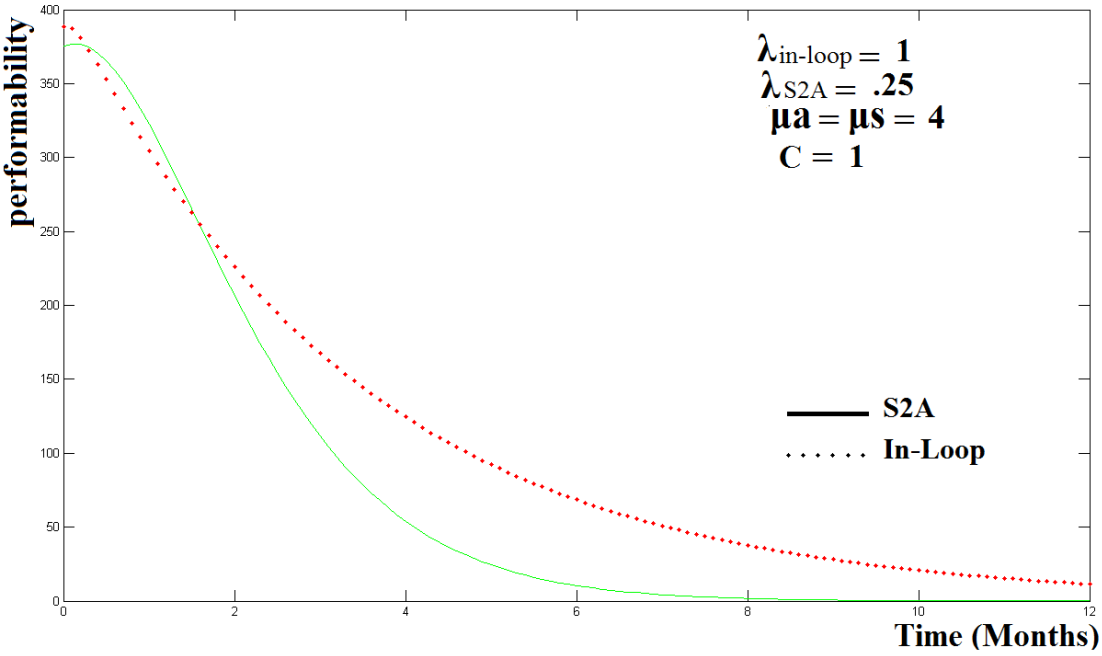

Figure 35: Performability S2A vs. In-Loop Model case study 6

Finally, The effect of coverage c will be studied and its effect on performability for both models as shown in the following case studies graphs where two value are chosen for the coverage c=.98, c=.96.

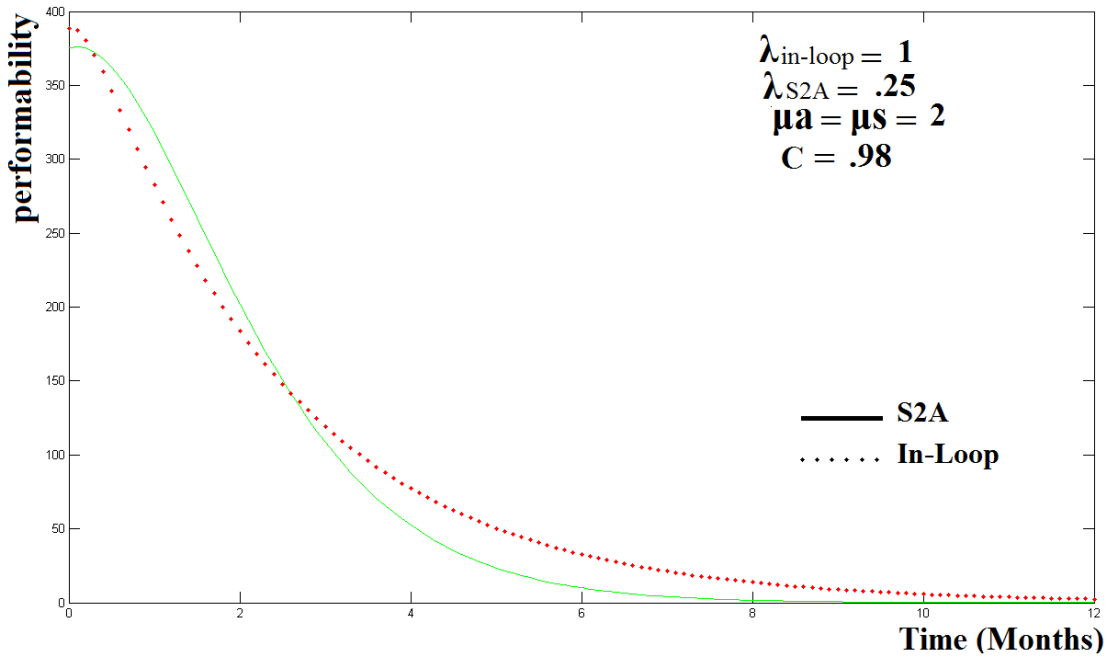

Figure 36: Performability S2A vs. In-Loop Model case study 7

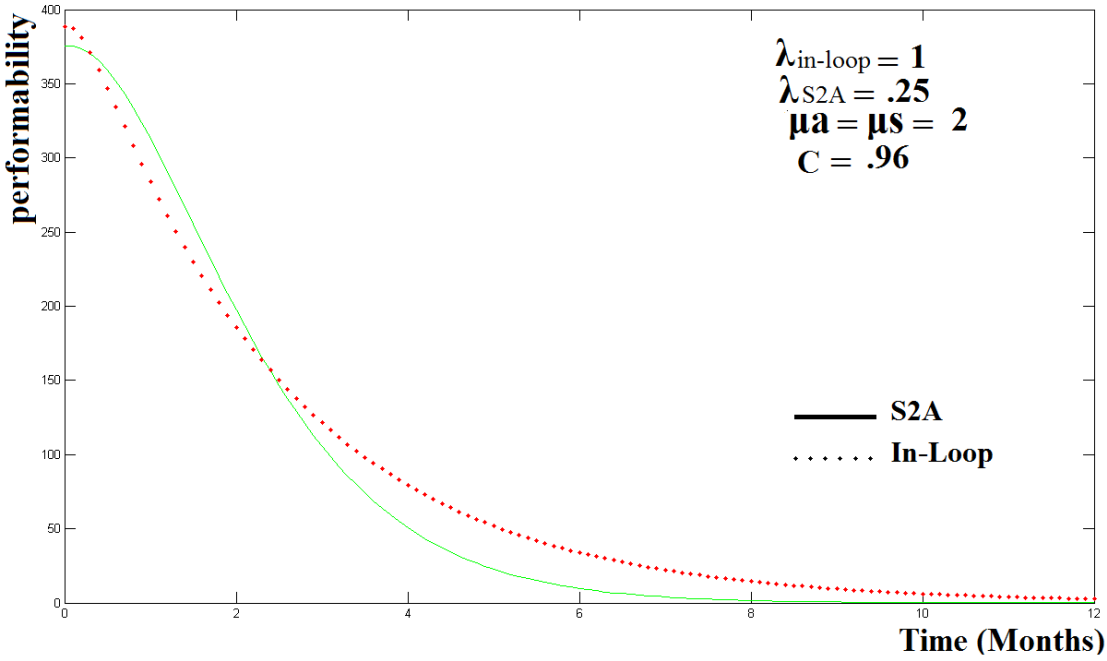

Figure 37: Performability S2A vs. In-Loop Model case study 8

It is found that the performability of both the S2A and the in-loop models does not change significantly with the change in coverage c. It can be concluded that small changes with coverage has a minor effect on the performability compared to the effect of failure and repair rates.

VI. CONCLUSION

Meeting time constraint requirement is a major system design requirement in Networked Control Systems (NCSs) for sensors, controllers and actuators loop. Different deterministic protocols were studied trying to maintain requirements of speed and correctness such as Controller Area Network (CAN) and PROFlBUS. Recently, Ethernet has appeared in the world of wired communication systems being one of the most widespread, familiar and low cost protocols available. Although Ethernet is a non-deterministic protocol by nature, researchers in academia and industry did not stop using the Ether-Channel as a communication medium for control systems and it was proved to be a very successful protocol. With the use of Ethernet, many things that were not possible in past implementations of NCS will be enabled such as interconnecting the industrial floor with the management floor. As a result, This will help in solving problems such as diagnostic and set-up and more functions can be added.

Therefore, a direct sensor to actuator architecture (S2A) was proposed in this research using unmodified switched Fast or Gigabit Ethernet to maintain low end-to-end delay. The delay measurements include all types of encapsulation/decapsulation, propagation and queuing delays. This architecture consists of 16 sensors, 4 actuators and a supervisor node. The supervisor is responsible for monitoring network behavior by exchanging packets with all the 20 different nodes in the network. It was shown that this proposed architecture was successful in meeting all required timing constraints with strict zero packet loss (with no over-delayed packets) requirements.

The S2A model was then compared to a traditional in-loop architecture with the same number of sensors and actuators; however this architecture has one controller

only. The in-loop controller receives packets from the sensors, calculate the control action and finally sends control packets to the actuators. Both architectures were studied for Fast and Gigabit Ethernet with and without additional load. Different scenarios were tested including different links bandwidth (Fast or Gigabit Ethernet) with or without additional load. The proposed S2A model showed better end-to-end delay results compared with the previous traditional model including in-loop controller without additional load.

Additional load was added to both models and it was modeled as a TCP application with different flat file sizes between the controller/supervisor and an external node to the networks which represents a maintenance engineer communicating with the controller/supervisor. This additional load increased the maximum end-to-end delay for the in-loop model, while did not affect the end-to-end delay of the S2A model using Fast and Gigabit Ethernet. Using Fast Ethernet, the in-loop model delay was increased exceeding the system deadline (which represents a system failure as it violates the delay constraints of real-time control systems). The end-to-end delay of the proposed model was not affected by the additional load because the supervisor node communicates in parallel with the network and with the external node. While, on the other hand, the delay in the in-loop model was increased because the controller node works in series with both the network and the external node.

Two different controller-level fault-tolerant models based on the in-loop and S2A models were also presented in this research. There are two separate controllers in the fault-tolerant in-loop model where one of them is active and the other one is in hot-standby mode. While in the fault-tolerant S2A model, two controllers will be integrated with an actuator in the same node. It was shown that both of the proposed

models were successful in meeting all required timing constraints and no packets were dropped using Fast and Gigabit Ethernet. However, the in-loop fault-tolerant model performed better in terms of less total end-to-end delay than the S2A fault-tolerant model in the fault-free situation. This is due to the fact that there are two separate controllers in the in-loop model compared with 8 controllers (two controllers per actuator) in the S2A model thus increasing the delay. On the other hand, in the scenario with the failed controller(s) (one controller in the in-loop model or one controller from each of the four pairs of integrated controllers in the S2A model), the S2A model was shown to have less total end-to-end delay. The traffic in the direct S2A model is sent over one hop compared to the in-loop model where the traffic must go through an additional intermediate hop via the controller thus increasing the experienced end-to-end delay. Different scenarios were tested including different link bandwidths (Fast or Gigabit Ethernet) in both the fault-free scenario and the scenario with the failed controller(s).

Finally, performability analysis between the two models was studied and compared using fast Ethernet links relating failure data with reward, depending on the system state. The reward was considered to be how far is the total end-to-end delay in each state in each model from the system deadline. The system deadline is taking as reference because failing to meet this deadline represents a failure of the system in Networked Control Systems. A case study was presented that simultaneously investigates the failure on the controller level with reward.

REFERENCES

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

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

[3] Official Site For PROFIBUS and PROFINET. http://www.profibus.com.

[4] Bosch. CAN Specification version 2.0 ISO 11898, 1991

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

[6] Official Site for Control Net. http://www.odva.org/Home/ODVATECHNOLOGIES/ControlNet.aspx.

[7] T. Skeie, S. Johannessen, and C. Brunner, "Ethernet in substation automation," IEEE Control Syst., Vol. 22, No. 3, 2002.

[8] IEEE 802.3 Standard.

[9] J.D. Decotignie, "Ethernet-Based Real-Time and Industrial Communications ", Proceedings of the IEEE, Vol. 93, No. 6 2005, pp.1102 - 1117.

[10] G. Marsal, "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.

[11] M.Felser, "Real-Time Ethernet – Industry prospective," Proceedings of the IEEE, vol. 93, no. 6, June 2005, pp.1118 - 1129

[12] F.L. Lian, J.R. Moyne, and D.M. Tilbury. "Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet," IEEE Control Systems Magazine, Vol. 21, No.1, February 2001, pp.66-83.

[13] B. Lounsbury and J. Westerman. "Ethernet: Surviving the Manufacturing and Industrial Environment," Allen-Bradley white paper, May 2001.

[14] S.-H. Lee and K.-H. Cho. "Congestion Control of High-Speed Gigabit-Ethernet Networks for Industrial Applications," Proceedings of the IEEE ISIE, Pusan, Korea, June 2001, pp. 270-275.

[15] J.S. Meditch and C.-T.A. Lea. "Stability and Optimization of the CSMA and CSMA/CD Channels," IEEE Transaction Communication, Vol. 31, No. 6, June 1983, pp. 763-774.

[16] P. Pedreiras, L. and Gai, P. Almeida. "The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency," Proceedings of the IEEE Euromicro Conference on Real-Time Systems ECRTS, Vienna, Austria, June 2002, pp.134-142.

[17] B.P. Upender, "Analyzing the Real-time characteristics of Class C communications in CAN through Discrete Event Simulations," Advanced Digital Systems, United Technologies Research Center.

[18] J. Ferreira, P. Pedreiras, L. Almeida, and J. Fonseca, “Achieving Fault-Tolerance in FTTCAN,” Proceedings of the IEEE International Workshop on Factory Communication Systems WFCS, Vasteras, Sweden, August 2002.

[19] K. Steinhammer and A. Ademaj, “Hardware implementation of the Time-Triggered Ethernet controller,” Embedded System Design: Topics, Techniques and Trends, Vol. 231.Springer Boston, 2007.

[20] “Overview of current automotive protocols,” Vector CANtech Inc., 2003, Available: www.vector-cantech.com.

[21] IEC 61784-2, Available at: www.iec.ch.

[22] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, “Diagnosis and Fault-Tolerant Control,” 2nd Edition, Springer, 2006.

[23] J. Nilsson, “Real-Time Control Systems with Delays,” PhD thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998, Available: http://home.case.edu/ncs

[24] F.-L. Lian, J.R. Moyne, and D.M. Tilbury, “Networked Control Systems Toolkit: A simulation package for analysis and design of control systems with network communication,” Tech. Rep.,UM-ME-01-04, 2001.

[25] R. Daoud, H. Elsayed, H. Amer and S. Eid, “Performance of Fast and Gigabit Ethernet in Networked Control Systems,” Proceedings of the 46th IEEE Midwest Symposium on Circuits and Systems MWSCAS, Cairo, Egypt, December 2003, pp.505-508.

[26] K. Tolly, “The Great Networking Correction: Frames Reaffirmed,” Industry Report, The Tolly Group, IEEE Internet computing, 1997.

[27] B. Wittenmark, B. Bastian, and J. Nilsson, “Analysis of Time Delays in Synchronous and Asynchronous Control Loops,” Lund Institute of Technology, 37th CDC, Tampa, Dec. 1998.

[28] J. Wang, and S. Keshav, “Efficient and Accurate Ethernet Simulation,” Cornell Network Research Group (C/NRG), Department of Computer Science, Cornell University.

[29] B. Moss, “Real-time Control on Ethernet,” *Dedicated Systems*, No. 00q2, April 2000, pp.53-60.

[30] R.M. Daoud, H.H. Amer, H.M. Elsayed and Y. Sallez, “Ethernet-Based Car Control Network*,”* Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering CCECE, Ottawa, Canada, May 2006, pp.1031-1034.

[31] IEC 61784-1 and 61784-2. www.iec.ch

[32] P.R. Kumar. “New Technological Vistas for Systems and Control: The Example of Wireless Networks,” IEEE Control Systems Magazine, vol. 21, no. 1, 2001, pp. 24-37.

[33] J.-P. Georges, “Systèmescontrôles en réseau: Evaluation de performances d’architectures Ethernet commutées,” PhD thesis, Centre de Recherche en Automatique de Nancy CRAN, France, Nov. 2005.

[34] B. Brahimi, “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, France, Dec. 2007.

[35] P. Marti, J.M. Fuertes and G. Fohler, "An integrated approach to realtime distributed control systems over fieldbuses," Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation ETFA, Antipes/Juan les Pins, France, October 2001.

[36] Z. Zinonos, R. Silva, V. Vassiliou and J.S. Silva, "Mobility solutions for wireless sensor and actuator networks with performance guarantees," Proceedings of the International Conference on Telecommunications (ICT), Ayia Napa, Cyprus, May 2011, pp.406-411.

[37] J. Cecilio, P. Martins, J. Costa and P. Furtado; , "A configurable middleware for processing in heterogeneous industrial intelligent sensors," Proceedings of the IEEE International Conference on Intelligent Engineering Systems (INES), Lisbon, Portugal, June 2012.

[38] S. Aoki, Y. Kirihara, J. Nakazawa, K. Takashio and H. Tokuda , "A sensor actuator network architecture with control rules," Proceedings of the Sixth International Conference on Networked Sensing Systems (INSS), Pittsburgh, USA, June 2009, pp.145-150.

[39] N. Rollins, M.J. Wirthlin, M. Carey, and P. Graham, "Evaluating TMR techniques in the presence of single event upsets," Proceedings of the 6th Annual International Conference on Military and Aerospace Programmable Logic Devices, Washington, USA, September 2003.

[40] R.M. Daoud, H.H. Amer, and H.M. ElSayed, "Gigabit Ethernet for redundant networked control systems," Proceedings of the IEEE International Conference on Industrial Technology, December 2004, pp.869-873.

[41] R.M. Daoud, H.H. Amer, and H.M. ElSayed, "Fault-tolerant networked control systems under varying load," SMCia/05. Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications, June 2005. pp.218-221

[42] M.Blanke, M.Staroswiecki and N.Wu, "Concepts and Methods in Fault-Tolerant Control," Proceedings of the American Control Conference, Arlington, VA, June. 2001, pp. 2608-2620.

[43] R.M. Daoud, H.H. Amer, and H.M. ElSayed, "Fault-Tolerant Two-Level Pyramid Networked Control Systems," Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation ETFA, Catania, September 2005, CF 000471, 6 pp. - 974.

[44] H.H.Amer, M.S.Moustafa and R.M.Daoud, "Optimum Machine Performance in Fault-Tolerant Networked Control Systems," Proceedings of the IEEE International Conference on Computer as a Tool, November 2005, pp. 346-349

[45] H.H.Amer, M.S.Moustafa and R.M.Daoud, "Availability of Pyramid Industrial Networks," Proceedings of the Canadian Conference on Electrical and Computer Engineering CCECE, Ottawa, May 2006, pp. 1862-1865.

[46] H.H.Amer and R.M.Daoud, "Parameter Determination for the Markov Modeling of Two-Machine Production Lines," Proceedings of the International IEEE Conference on Industrial Informatics INDIN, Singapore, August 2006,

pp. 1178-1182.

[47] Thomsen, J.S. and Blanke, M. "Fault-tolerant Actuator System for Electrical Steering of Vehicles." Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society IECON, Paris, November 2006, pp. 3597-3602.

[48] Zhang Jianjun, "Fuzzy robust fault-tolerant control for networked control systems." Proceedings of the International Conference on Computer Science & Education ICCSE, Melbourne, July 2012, pp.1267-1270.

[49] T.K.Refaat, R.M.Daoud, and H.H.Amer, "Fault-tolerant controllers in Wireless Networked Control System using 802.11g," Industrial Technology (ICIT), IEEE International Conference on , vol., no., pp.783,788, 19-21 March, 2012.

[50] B.R. Haverkort, R. Marie, G. Rubino, and K. Trivedi, "Performability modelling: techniques and tools," John Wiley & Sons, 2001.

[51] Official Site For OMNeT++: www.omnetpp.org/

[52] L. Seno, S. Vitturi, and F. Tramarin, "Experimental Evaluation of the Service Time for Industrial Hybrid (Wired/Wireless) Networks under Non-Ideal Environmental Conditions," Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation ETFA, Toulouse, France, September 2011, pp.1-8.

[53] Official Site For SIEMENS: http://www.siemens.com/entry/cc/en/

[54] G. Boggia, P. Camarda, V. Divittorio, and L.A. Grieco, "A simulation-based performance evaluation of Wireless Networked Control Systems," Proceedings of the14th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, Mallorca-Spain, September 2009, pp.1-6.

[55] J.F. Kurose and KW. Ross, *Computer Networking:* A Top-Down Approach Featuring the Internet. Addison Wesley Publishing Company, 2000.

[56] Moustafa, E.A, Halawa, H.H, Daoud, R.M. and Amer, H.H, "Sensor Actuator Ethernet-Based Networked Control Systems," Proceedings of the 14th International IEEE Conference on Sciences and Techniques of Automatic Control and Computer Engineering STA, Sousse, December 2013, pp.530,534.

[57] D.P. Siewiorek and R.S. Swarz, Reliable Computer Systems – Design and Evaluation, A K Peters, Natick, MA, USA, 1998.

[58] K.S. Trivedi, Probability and statistics with reliability, queuing, and computer science applications, Wiley, NY, USA 2002.

[59] T.F. Arnold, "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, pp.251-254.

[60] H.H. Amer and E.J. McCluskey, "Calculation of Coverage Parameter", IEEE Trans. Reliability, June 1987, pp. 194-198.

[61] D.P.Siewiorek and R.S.Swarz, "The Theory and Practice of Reliable System Design."Digital Equipment Corporation, 1982.