Fault-tolerant NCSs with video sensors

Hadeer Ahmed
The American University in Cairo

Follow this and additional works at: https://fount.aucegypt.edu/etds

Recommended Citation

APA Citation
https://fount.aucegypt.edu/etds/1500

MLA Citation
https://fount.aucegypt.edu/etds/1500

This Master’s Thesis is brought to you for free and open access by the Student Research at AUC Knowledge Fountain. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AUC Knowledge Fountain. For more information, please contact mark.muehlhaeusler@aucegypt.edu.
Fault-Tolerant NCSs with Video Sensors

A Thesis Submitted to

Electronics and Communication Engineering Department

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

by Hadeer Ahmed AbdelHamid Mohamed Ahmed

under the supervision of Prof. Hassanein H. Amer and Dr. Ramez Daoud

July 2019
This page is reserved for the approval sheet.
DEDICATION

I dedicate this thesis to my grandma, may her soul rest in peace, I miss you so much. I also dedicate it to my grandpa, he’s my role model, I learned a lot from him, I look up to being like you.

I dedicate this thesis to my parents; Mum: I am who I am because of you, your endless love, care and effort were my support and motivation throughout the years, Dad: you’re the best dad anyone can ask for, thank you so much for the love, care & sacrifice. Mohamed: you’re the kindest and best brother ever.

I would also like to dedicate this thesis to my small family. Tarek: Thank you so much for being in my life, your love and support through the Masters were my greatest motivation. Omar: my baby, my life has changed because of you. You are the best gift ever; may you fulfill all your dreams.

I would like to thank Prof. Hassanein Amer for his support and care throughout my undergraduate and graduate studies, you’re my mentor.

I would like to thank my all friends, especially Nada for her support & faith.

Finally, I would like to say that words can’t express how grateful I am to be part of my family and to be surrounded by such caring people. Thank you, I love you all.
ACKNOWLEDGEMENTS

I would like to acknowledge Prof. Hassanein Amer, for his endless support, patience and above all care. I am forever grateful for everything. I learnt a lot from him, he’s my mentor.

I would also like to acknowledge Dr. Ramez Daoud for all his help & support during my thesis.

I would also like to acknowledge Dr Ihab Adly and Prof. Hany EL-Sayed for their support and advise throughout the journey.

I would finally like to acknowledge and thank Eng. Gehad Al-Kady, Eng. Tarek Refaat, Eng. Hassan Halawa & Eng. Manar Negm for all their continuous support and assistance throughout my thesis. The work wouldn’t have been completed without their help. Thank you.
# Table of Contents

Chapter 1 ............................................................................................................................................. 1

Introduction ........................................................................................................................................... 1

1.1 Background ...................................................................................................................................... 1

1.2 Contribution of this Thesis .............................................................................................................. 2

1.3 Thesis Organization ......................................................................................................................... 3

Chapter 2 ............................................................................................................................................. 4

Literature Review ................................................................................................................................. 4

2.1 Networked Control Systems (NCS) ................................................................................................. 4

2.2 Fault tolerance ................................................................................................................................. 13

2.3 Previous work on Fault tolerance in NCS ....................................................................................... 16

2.4 FPGAs ............................................................................................................................................. 17

2.5 Previous work on Fault tolerance on FPGAs ................................................................................ 21

Chapter 3 ............................................................................................................................................. 24

Performability of Fault-Tolerant NCSs with Video Sensors .................................................................... 24

3.1 Proposed Model ............................................................................................................................... 24

3.2 Riverbed Simulations ....................................................................................................................... 27

3.3 Simulation Results ........................................................................................................................... 33

3.4 Performability Analysis ................................................................................................................... 36

Chapter 4 ............................................................................................................................................. 43

Reliable FPGA-based Camera Sensor for NCS .................................................................................... 43

4.1 Modified model ............................................................................................................................... 43

4.2 Reliability Modeling ......................................................................................................................... 45

4.3 Case Study ..................................................................................................................................... 53

4.4 Summary ....................................................................................................................................... 55

Chapter 5 ............................................................................................................................................. 56

Conclusions and Recommended Future Work ..................................................................................... 56

References ............................................................................................................................................ 59

List of Publications from the Thesis .................................................................................................... 65
## List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Industrial Revolution [1]</td>
<td>1</td>
</tr>
<tr>
<td>2.1</td>
<td>NCS</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Transmission Procedure of CSMA/CD [13]</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>An example of traffic shaping into three traffic priorities</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Comparison of transmission methods of Ethernet and switched Ethernet [13]</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Bathtub curve; failure rate evolution of a system [17]</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>Exponential failure rate [17]</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>FPGA Architecture [28]</td>
<td>18</td>
</tr>
<tr>
<td>2.8</td>
<td>Partial Reconfiguration [31]</td>
<td>19</td>
</tr>
<tr>
<td>2.9</td>
<td>Fault Secure TMR voter Architecture</td>
<td>21</td>
</tr>
<tr>
<td>2.10</td>
<td>Implementation of majority function voter on FPGAs [39]</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>General model</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Conveyor belt</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Riverbed Simulation Model</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>K failure Noiseless Model</td>
<td>28</td>
</tr>
<tr>
<td>3.5</td>
<td>KV failure Noiseless Model</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>fault free Noisy Model</td>
<td>31</td>
</tr>
<tr>
<td>3.7</td>
<td>failure Noisy model</td>
<td>32</td>
</tr>
<tr>
<td>3.8</td>
<td>KV failure Noisy Model</td>
<td>33</td>
</tr>
<tr>
<td>3.9</td>
<td>Packet End-to-End Delay of K in FF noiseless Scenario</td>
<td>35</td>
</tr>
<tr>
<td>3.10</td>
<td>Markov Model</td>
<td>37</td>
</tr>
<tr>
<td>3.11</td>
<td>Performability for the 300KB, 768 &amp; 900KBs camera for 16 hours.</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>FPGA Controllers</td>
<td>44</td>
</tr>
<tr>
<td>4.2</td>
<td>An example of system behavior in the case of KV1 failure</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>Transitions from state &quot;00000&quot;</td>
<td>47</td>
</tr>
<tr>
<td>4.4</td>
<td>Transitions from state &quot;10000&quot;</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Transitions from state &quot;01010&quot;</td>
<td>49</td>
</tr>
<tr>
<td>4.6</td>
<td>Markov Model on SHARPE</td>
<td>52</td>
</tr>
</tbody>
</table>
List of Tables

Table 3.1 End to End delay for noiseless Scenarios ................................................................. 34
Table 3.2 End to End delay for noise Scenarios ................................................................. 35
Table 3.3 Reward for noiseless scenarios .............................................................................. 40
Table 4.1 Down States ........................................................................................................... 51
Table 4.2 Comparison Between R_modified, R_previous & R_simplex .................................. 54
LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Actuator</td>
</tr>
<tr>
<td>AV</td>
<td>Video Actuator</td>
</tr>
<tr>
<td>CLB</td>
<td>Configurable Logic Block</td>
</tr>
<tr>
<td>DPR</td>
<td>Dynamic Partial Recovery</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FT</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial, Scientific and Medical</td>
</tr>
<tr>
<td>K</td>
<td>Controller</td>
</tr>
<tr>
<td>KARM</td>
<td>ARM Controller</td>
</tr>
<tr>
<td>KV</td>
<td>Embedded video controller</td>
</tr>
<tr>
<td>KV1</td>
<td>Video controller 1</td>
</tr>
<tr>
<td>KV2</td>
<td>Video controller 2</td>
</tr>
<tr>
<td>KV3</td>
<td>Video storage controller</td>
</tr>
<tr>
<td>LUT</td>
<td>Lookup table</td>
</tr>
<tr>
<td>MBU</td>
<td>Multiple Bit Upset</td>
</tr>
<tr>
<td>MTBF</td>
<td>Mean time between failures</td>
</tr>
<tr>
<td>MTTR</td>
<td>Mean time to repair</td>
</tr>
<tr>
<td>MTTF</td>
<td>Expected time of failure</td>
</tr>
<tr>
<td>NCS</td>
<td>Networked Control System</td>
</tr>
<tr>
<td>S</td>
<td>Sensor</td>
</tr>
<tr>
<td>SBE</td>
<td>Single Bit Error</td>
</tr>
<tr>
<td>SEEs</td>
<td>Single Event Effects</td>
</tr>
<tr>
<td>SEFI</td>
<td>Single Event Functional Interrupt</td>
</tr>
<tr>
<td>SEL</td>
<td>Single Event Latch up</td>
</tr>
<tr>
<td>SET</td>
<td>Single Event Transient</td>
</tr>
<tr>
<td>SEUs</td>
<td>Single Event Upsets</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SV</td>
<td>Video Sensor</td>
</tr>
<tr>
<td>S2A</td>
<td>Sensor-to-Actuator</td>
</tr>
<tr>
<td>UDP</td>
<td>Universal Datagram Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TDDDB</td>
<td>Time-Dependent Dielectric Breakdown</td>
</tr>
<tr>
<td>TMR</td>
<td>Triple Modular Redundancy</td>
</tr>
</tbody>
</table>
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>Failure rate</td>
</tr>
<tr>
<td>$R(t)$</td>
<td>Reliability at time t</td>
</tr>
<tr>
<td>$TP(t)$</td>
<td>Transient Performability</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Repair rate</td>
</tr>
<tr>
<td>$Rew$</td>
<td>Reward</td>
</tr>
<tr>
<td>$S$</td>
<td>Temporary failure rate of soft core</td>
</tr>
<tr>
<td>$H$</td>
<td>Permanent failure rate of soft core</td>
</tr>
<tr>
<td>$Hk$</td>
<td>Permanent failure rate of the controller</td>
</tr>
<tr>
<td>$Harm$</td>
<td>Permanent failure rate of the arm</td>
</tr>
<tr>
<td>$Z$</td>
<td>Conditional probability that a failure is temporary</td>
</tr>
</tbody>
</table>
Abstract

Networked control systems, NCS, consist of smart nodes: sensors, actuators, and controllers. The use of such smart nodes is essential in smart factories to be able to deal with Industry 4.0. NCS are widely implemented in lots of critical industrial applications. A main advantage of NCS is the abundant bandwidth. This research presents NCS that utilizes switched Ethernet with 16 sensing nodes, 4 actuation nodes & a smart controller. The model also consists of a smart sensor that has an embedded video controller and its actuator. The smart sensor processes the video captured by the onboard camera and the entrenched controller. The study added fault tolerant on the system where the sensor’s embedded controller acted as a spare controller to the system’s main and a performability analysis was studied. This system was simulated on Riverbed and various scenarios were tested in the presence and absence of noise. Simulations clearly indicated zero packet loss and no over-delayed packets. The noise in this research was an FTP file being exchanged between the noise node and the main controller. Later in the research, a modification on the system was done and an FPGA was utilized in the system. Making use of its Dynamic Partial Reconfiguration feature where some failures are being repaired without system interruption. In addition to DPR, FPGAs are powerful chips and are used progressively in industry. The modified model includes image storing feature and fault tolerant methods to increase the system’s reliability and lifetime. In addition to the mentioned features, the new system guarantees less traffic on the network in all cases in comparison to the previous model. Markov Models were developed for both systems and simulated using SHARPE package. A case study was presented to compare both systems in terms of reliability; results indicated that the modified system is more reliable.
Chapter 1

Introduction

1.1 Background

Over the years the industry has evolved. The introduction of water and steam-powered mechanical production is considered the first industrial revolution facilities in the late 1700s. Second, came the introduction of the electricity-powered assembly line that facilitated the mass production of many products. The development of PLCs in late 1960s initiated the third industrial revolution since, it enabled production automation. The massive development that is being currently in action is considered the latest industrial revolution, in other words; Industry 4.0.

Industry 4.0 transforms the previous-style manufacturing into smart manufacturing and forms the basis of the future smart factories as they grow more automated and intelligent [2]. Likewise, in industrial networks technologies such as fieldbus and Wireless Sensor Networks.
(WSN) proved their importance [3]. This is due to the importance of the network layer in monitoring and controlling the smart factory. The main physical resources of smart manufacturing resources are the reprogrammable industrial units, reprogrammable manufacture lines and smart data acquisition units [3]

Networked Control Systems (NCSs) consists of networked smart controllers transmitting data with smart sensors and actuators. Small frequent control packets are being communicated over the network. The traffic is sent to the controllers from the sensors. Controllers then, transmit the required action to the actuation nodes [4]. NCS decreases the system’s complexity and allows sharing of data since, the data is always accessible on the network. Thus, NCS can be of a vital role in factory automation. The addition of Fault Tolerance techniques to NCSs is important in increasing the system’s reliability and/or performability. Field Programmable Gate Arrays (FPGAs) are being currently used in numerous industrial control applications due to its flexibility and ability to recover impermanent errors resulted from the surroundings.

1.2 Contribution of this Thesis

This thesis studies a distributed NCS, composed of smart sensing nodes along with smart actuation nodes and smart controllers. At the controller level, fault-tolerance is applied. Network performance is assessed according to the end-to-end delays. The suggested model is demonstrated and tested via Riverbed Network Modeler [5], both in the fault-free scenario and the fault-tolerant scenarios. The system performance is also assessed in the existence and absence of a noise node that adds non-control traffic to the network. Then, a Performability analysis is presented and applied with a delay-based reward.

Modifications were then made to the previously presented architecture in order to solve the two issues. First, the video data is communicated on the network once the embedded video controller fails; which results in network congestion and is anticipated to cause an increase in the end to end delay particularly upon the increase in the frame size. Second thing is the storage issue; the captured video data cannot be retrieved later as, there was no storage node which can store this information. Since, some applications might entail saving the sensed images and transmitting them over the network every now and then to the supervisor for backup. Thus, the
key change is the addition of an FPGA and removing the entrenched video controller. The use of FPGA caused some additional cost. An Embedded dual core ARM processor is included in the FPGA chip. Fault Tolerance will be applied to the FPGA in to increase the system’s reliability. A comparison between both systems will be presented. The used of an FPGA will solve the storage issue. It should be mentioned that this research is concerned with the failure of each block in the FPGA (as in [6]) and not the total chip failure.

1.3 Thesis Organization

Chapter 2 includes the literature review in which it explains the concept of NCS in terms of functionality, advantages and applications. It also presents the architecture and the application of the previously presented NCS works. Then, the chapter explains some fault tolerant techniques used for NCS. It also presents the different methods that can be used to assess the fault tolerant technique used. Later in this chapter, previously used fault tolerance systems for NCS are shown. Finally, Field Programmable gate arrays are discussed in detail: operation, advantages and examples from the literature.

Chapter 3 presents the architecture of the proposed fault tolerant NCS with video sensors model. The system is explained in detail in terms of the sampling period, control packets, protocol and video used. The fault tolerant technique used in the proposed model was described. Then, the performability analysis explicated. Simulations done on the systems to assess the functionality and performability were also clarified. At last, the simulation results are presented.

Chapter 4 presents the modified architecture of the previously mentioned model in the earlier chapter. The modified model uses an FPGA. The fault tolerant technique used in this improved model is presented and the system reliability is computed. In addition, a comparison between both systems is carried and shown in a case study.

Lastly, the work is concluded in chapter 5.
Chapter 2

Literature Review

2.1 Networked Control Systems (NCS)

2.1.1 Introduction

NCS is composed of smart sensing nodes and actuation nodes communicating with smart networked controllers. Sensed data is transmitted to controller every sampling period. The controller is in charge of processing the data, taking control decisions & transmitting the actions to the actuators as shown in Fig. 2.1. It should be noted that the controls deadlines are met in the worst delay conditions. There are various industrial applications that uses NCS.

NCS are either event-triggered or time-triggered this is determined according to the availability of the clock signal [7]. In Time-triggered systems, samples are taken at discrete time with constant sampling period. On the other hand, in Event-triggered systems actions are taken when a certain event takes place while the sampling is continuous. NCS can be used in lots of real time and non-real time control applications.

Smart sensors/ sensing nodes are characterized by the capability to sense certain property such as motion, temperature, pressure and lots of others, and communicate with other nodes. In other words, it is responsible for data acquisition and communication. Intelligence in sensors
makes them capable for functioning independently. Sensed data can be locally processed. In some cases, depending on the application requirements, they are able to apply self-calibration/diagnosis. In addition, sensors correctly encode the output before communicating the data on the network [8] [9].

On the other hand, smart actuators play the same role as the smart sensors, but in the opposite way. To illustrate, they are in charge of decoding and post-processing of data if needed within the actuation node before applying the necessary action to the physical system [8].

Networked controllers are in charge of generating control commands and transmitting it to the actuators after processing the receiving the sensed data [8]. The control algorithm handles both centralized and decentralized information analysis. In a distributed, decentralized NCS, the control is divided among the nodes: smart sensing nodes, controllers and the smart actuation nodes. The network should always guarantee that delays are within the limits. Another important use of the controller nodes is the possibility of supporting human-machine interface to either operators or higher-level managers [8].

2.1.2 Advantages of NCS [10]

1. Flexibility; allows network extension and sharing.

2. Reduced Complexity

3. Data Sharing; data is available on the network nodes when needed.

4. Elimination of unnecessary wiring to build the large control system.

5. Extendibility; it is possible to add sub control systems and extend the network to be connected to the central control system without massive changes to the physical layout of the system.

6. Remotely controlled
2.1.3 Protocols

Fieldbus or in other words, the real-time industrial network, is a central component for building automated industrial structures. Protocols should be more intelligent and flexible. Accordingly, to be able to meet the real-time necessities and restrictions required by NCS devices, various standard organizations and retailers have developed several fieldbus protocols.

Protocols such as: WorldFIP, Profibus and Foundation Fieldbus, are declared as global standard and are widely used over Ethernet because of their deterministic characteristics. However, these protocols are limited in throughput. Besides the business market for such protocols is limited for the following reasons. First, these protocols are specific to applications. Second, for applications that require higher quality information it is hard to upgrade due to the high price of hardware and complexity of interfacing with different vendors products. EtherCAT is a protocol that provides high real-time performance; the master is implemented with standard components while the slave requires special hardware to maintain the short packet forwarding delays in the devices. ModBus TCP is a protocol that supports the communication between smart devices on serial line, TCP was used for the data transport and is compatible with other Ethernet products. Profinet is the successor of Profibus, it offers real time support and allows the integration of fieldbuses. Ethernet is more convenient, available off shelf and cheaper along with it gives high speeds that is expected to increase even more by time. Hence, Ethernet is used in the industrial automation area after enhancing the predictability by applying some modifications.

Ethernet

More than 40 years ago in the early 1970s at Xerox, Ethernet was developed by Metcalfe to connect a PC to a laser printer. The IEEE 802.3. standardized the Ethernet but some minor differences to the original format. 2.94Mbps was the starting transmission rate for the protocol till it reached the current speed of 1Gbps and 10Gbps and is expected to increase. Throughout the years and by use of hubs and unshielded twisted pair cables, the physical medium changed from a thick coaxial cable bus to a star topology [11].

Since Ethernet is fast and easy to install with low-priced ICs’ interfaces, nowadays Ethernet is best solution for local area networks at homes and offices. Consequently, due to that
wide-spread usage, Ethernet interfaces are being installed in almost all electronic equipment [12]. The Ethernet was originally invented for local usage as mentioned before, currently it’s being proposed and implemented in industrial applications [11] [12].

The MAC for Ethernet is CSMA/CD and is illustrated in Fig.2.2. For a frame to be transmitted, transmission medium should be checked whether it is busy or not by the source station. The source denies transmission and pauses till the medium converts idle, if the medium is jammed. Else if the medium is idle, the source sends the frame. A jam signal is sent to alert an incidence of the collision to other stations when a collision is perceived during the transmission and the source interposes the transmission and broadcasts. After collision, a backoff time is paused depending on the truncated binary exponential backoff (BEB) algorithm and then resends. This retry can be done not more than 16 times, after that the transmission is inhibited. CSMA/CD algorithm causes the communication delay to be comparably high under heavy traffic due to frequent collisions. The network throughput can decrease significantly due to discarding the frames after 16 unsuccessful transmission attempts this happens in extreme cases [13].
An example will be explained to further clarify the non-determinism issue in Ethernet. Assume a message is to be transmitted by two stations, while the medium is busy being used by a third station to transmit its message. Once the third station has finished transmitting, only one of the two awaiting stations will be able to successfully transmit its message. Knowing in advance which of the two stations is transmitting its message first is impossible. Hence, an upper bound to the time needed to send a message from one node to the other cannot be specified. The only option is to evaluate the probability that this time will not surpass a certain value. This randomness in time determination was not accepted because real-time guarantees were needed by the users. [12].

Figure 2.2 Transmission Procedure of CSMA/CD [13]
Some of the main industrial communications features are: Time-deterministic communication, Exchange of very small data record in a frequent and efficient manner, and Timely synchronization of actions between the field devices [14]. Since industrial applications are Real-time based, it requires the mentioned features. Consequently, some modifications have been made to Ethernet to achieve the required behaviour. Below are few examples:

1. Modifications in MAC
   In this approach, changes are applied in the MAC layer. By this access time to the bus is limited. For instance, a categorized priority of nodes can be set. Thus, if a collision takes place, the nodes with the lowest priority automatically stop competing for the bus unlike the nodes with the higher priority. These nodes will keep sending till a successful transmission happens. There are two main downsides in this approach. First, is that changes have to be applied to the firmware increasing the cost of Ethernet and losing one of its core advantages. Second, the maximum transmission time can be a lot higher than the regular transmission time and this a major disadvantage, as transmission time is one of the crucial elements to real-time applications. Hence, such approach isn’t widely spread [11].

2. Adding transmission control over Ethernet:
   In this methodology, an extra layer is implemented on top of Ethernet as an approach to accomplish real-time-constrained applications over Ethernet. The responsibility of this layer is to control message transmissions timing. By this, either an upper bound of the number of collisions is achieved or the collisions are prevented altogether. The advantage of using this approach instead of the modified the MAC layer method is that the hardware doesn’t change. Consequently, the standard Ethernet hardware can be used that is of low cost. Master/Slave, Token-Passing, TDMA are some of the methodologies that can be done to achieve this control over Ethernet, and each of which has its pros and cons [11].
3. Traffic Shaping:

In this solution, the main concern is to reduce the probability of collision by reducing the bus utilization. Yet, this is just reducing the number of collisions and not eliminating it. Thus, a given probability of collisions on the bus can be concluded, if the traffic on the bus is set below a certain threshold and bursts of traffic are prevented. As a try to attain this, for example, traffic smoother, an interface layer, is employed between the Ethernet layer and the transport layer (TCP or UDP). As previously discussed, Real-time traffic is event-triggered; hence, it is communicated on demand and spread on the time domain. Unlike non-real-time traffic, it is supposedly transmitted in bursts and is handled by the traffic smoother. The function of the smoother is to track the previous transmissions of messages executed by the node. Knowing the transmission history of the node and according to its assigned transmission rate, the non-real-time traffic is transmitted in a controlled manner. Therefore, at the node level, the transmission of real-time messages gets a higher priority in comparison to the transmission of non-real-time messages. At the network level, a probabilistic bound is set to the interference due to non-real-time traffic [11].
4. **Switched Ethernet**

![Diagram of Ethernet and Switched Ethernet](image)

*Figure 2.4 Comparison of transmission methods of Ethernet and switched Ethernet [13].*

The switched Ethernet is the promising industrial fieldbus protocol that is different than the conventional Ethernet in several features. The first change is in operation mode of the hub in which all the nodes are connected. The hub is a passive device as shown in Fig.2.3a. It repeats the received data from the input port to all output ports in other words, it acts as a broadcaster. Unlike in the Switched Ethernet, the hub is an active device which recognizes and transmits the frame only to the destination as presented in Fig. 2.3b. Consequently, frames’ collision can be avoided if destinations of the transmitting multiple stations are different. The second change is the connection type between a station and the hub. The switched Ethernet uses a full-duplex link; the station can transmit and receive frames simultaneously, while the conventional Ethernet uses a half-duplex link. This allows a node to send and accept frames simultaneously. Thus, because of the two major changes, the switch ethernet can be considered free of collisions [13].
2.1.4 Previous works on NCS

Many control applications work at relatively low speeds in comparison to the speeds of the new network standards. Without handling the control packets in a special manner, the required delays of these packets can be obtained under realistic loading conditions. Various control applications run at relatively lower speeds in comparison to the that of the new network standards. Without handling the control packets in a special manner, logical loading conditions could be the reason for reaching the required delays of these packets. In [9], two models were modeled to study whether systems that fail with Fast Ethernet switches can work properly with Gigabit Ethernet. For the sake of performance comparison, one model is simulated on-top-of Fast Ethernet and the other one is simulated over Gigabit Ethernet. The light traffic system model is represented by 16 sensing nodes, one controlling element and 4 actuation nodes; this is based on the model implemented in [15]. One the other hand the heavy traffic model entails 48 sensing nodes, one controlling element and 4 actuation nodes. The results of this study validate the concepts that while using high speed Ethernet networks, standard switches accommodate the timing requirements of several control systems. Simulations presented that if this traffic is within practical limits, the control packets will not be affected in the case of additional traffic resulting from integration of other functions. It may require to be limited at the upper layer software instead of at the switch itself in order to possess the non-real-time traffic within practical limits.

In [16], the study was focused on light traffic NCS where the network consisted of 16 sensing nodes, one controlling element and 4 actuation nodes connected in star Ethernet. The machine is adjusted to run at one revolution per second for 60 strokes per minute. The sampling frequency was adjusted to be 1440 Hz. Consequently, 28,800 packets are being processed per second. This is equal to multiplying the count of actuation nodes and sensing nodes by the sampling frequency. Adding non-real-time traffic of: FTP, HTTP, telnet and e-mail check was the applications used to simulate the operator intervention on the machine. In case of two machines operating with one controller, it is advised to block non real time traffic. The blockage is done by discontinuing the FTP access at the application layer under this serious operation. At last, the study concluded that when applying the heavy composite load (real-time and non-real-time) to the simulated networks neither Ethernet speeds can tolerate. Alternatively, Fast Ethernet
failed to accommodate the real-time traffic and to send packets within the restricted time frame unlike the Gigabit Ethernet that successfully did.

2.2 Fault tolerance

The basic definition for fault tolerance is the proper and correct system operation in the presence of fault [17]. The importance of fault tolerance is crucial in safety applications to avoid problems. Nowadays, systems became more complex and as the complexity of the system increases, the reliability decreases. In addition, faults can occur for any reason even if the system was tested several times and the hardware were checked and correctly operating. Therefore, the necessity of fault tolerant system increased. This is because fault tolerant systems ensure the correct operation of the system even with some faulty parts. An important aspect in fault tolerant system is dependability where the system is able to perform its planned level of performance [17].

All forms of Performability: Steady State, Cumulative and Transient Performability (SSP, CP and TP respectively) can be analyzed [18]. Reliability Modeling is one of the various techniques that can quantify the increase in system’s reliability. Basically, the probability that the system functions without failure in the interval [0, t], given that the system is working correctly at time (t = 0), is known as Reliability R (t). In other words, Reliability represents the probability of success and varies with time.

The Failure rate, λ, of any system is the expected number of failures per unit time. The below curve, Bathtub curve in Fig. 2.2, presents the typical failure rate evolution of a system over its lifetime.
The curve is divided into 3 main phases: infant mortality, useful life and wear out. To start with, the infant mortality phase is when the system is at the start of its life cycle. In this phase, $\lambda$ sharply decreases because of the weak components’ failures. As for the useful lifetime phase, $\lambda$ stabilizes and remains constant. In the last phase, the components begin to damage; increasing the failure rate.

In the useful lifetime phase, the system’s reliability decreases exponentially with time since the failure rate is constant; exponential failure law.

$$R(t) = e^{-\lambda t} \quad (1)$$

The below Fig. 2.3 demonstrates the law.
To determine the dependability of fault tolerant systems the following procedures are used such as Mean Time to Failure (MTTF), Mean Time to Repair (MTTR) and Mean time between Failures (MTBF).

**MTTF** is the expected time of the occurrence of the first system failure.

$$MTTF = \frac{1}{\lambda}$$  \hspace{1cm} (2)

**MTTR** is the average time required to repair the system.

$$MTTR = \frac{1}{\mu}$$ \hspace{1cm} (3)

**MTBF** is the average time between failures of the system and is equal the sum of MTTF and MTTR if the repair makes the system perfect.

$$MTBF = MTTR + MTTF$$ \hspace{1cm} (4)

Fault tolerant controls (FTCs) or in other words the fault tolerance techniques added to NCS is of crucial importance. FTCs can automatically get over components’ failure. One of the widely used FT techniques to NCSs is Modular redundancy. This redundancy technique improves the failure rate; in case of the combined component it is applied. An important advantage of Fault tolerance is the prevention of fault propagation throughout the system. Failure rate enhancements along with fault propagation prevention are critical for NCS applications especially safety-critical ones such as: aerospace, automotive, manufacturing and other process industries [19]. The increased safety and reliability of FTC systems resulted in wider range of applications that require implementation.

One of the most commonly encountered and widely implemented fault tolerance techniques in NCS is, Modular redundancy. It can be applied to any component in the system sensor, actuator and/or controller. Also, it can be applied on the network level. Triple Modular Redundancy (TMR) and Sift-Out Modular Redundancy are most popular techniques.

Although Fault detection techniques don't prevent or tolerate the error, it alerts the system when there is a faulty result. On the other hand, fault masking adds redundancy for fault tolerance. It either isolates the faulty results or isolate/prevent them from reaching the output of
the module. This technique isn’t dynamic, as element stays there with no actual intervention. Hence, once the masking redundancy is drained then the error will reach output [20].

The difference between failure, fault and error should be highlighted. Failure indicates a physical break where the output is always wrong. As for fault, it is the mathematical model that described the physical break. Error is basically an incorrect piece of information.

2.3 Previous work on Fault tolerance in NCS

In [21], the proposed model is basically a grouping of the S2A architecture that is described in [22] and the In-Loop architecture that is detailed in [4] connected through an Ethernet switch. The S2A is implemented for video sensor network application on the same FPGA board where the controller is embedded in the sensor node. The In-Loop architecture contains sensing nodes, controller and actuation nodes. The simulation setting is set to be very harsh with high probability of impermanent failures in FPGAs. This model requires high reliability as video sensors are very sensitive with rapidly changing input data. Both Hot standby [23] and Sift-Out which is based on organizing the system into L identical modules [24] fault tolerant techniques were investigated. The Single Event Upsets fault model was used and the Dynamic Partial Recovery (DPR) was used for reposssession. Reliability is then calculated after applying each fault tolerant method. It was shown that the amount of redundancy in the Sift-Out technique is much more than that implemented in Hot Standby. Thus, it is predicted to cause a more reliable yet more expensive system [20]. Markov Models were implemented for reliability calculations.

In [25], two and three-cell systems were studied. Each cell system contained sensing nodes, actuation nodes and a controller. Sensors transmit their data to an access point that is connected to a controller. The controller’s reply is sent to actuators through a similar link. All sensors send their data to all the controllers. Packet delay varies according to the number of operational controllers. The robustness of the system is dependent on the difference between the measured packet delay and the 16ms benchmark; In other words, the higher the difference, the more robust the system will be. In the case of controller failure regardless the count, the operating controllers take over the load. The network can accommodate with only one functional controller. Fault-tolerance is conducted in the presence and absence of noise. The noise was
added to neighboring ISM channels. At the end, a Performability analysis of the system was studied and another reward was assigned.

2.4 FPGAs

FPGAs are pre-fabricated silicon devices. They are considered a type of integrated circuits that are used to apply any digital circuit or system. Several applications in the fields of data processing, networks, industrial, space and automotive can be implemented using FPGAs. In addition, they are used in various industrial control applications such as FPGA-based speed controllers [26] and FPGA-based motor controllers [27].

2.4.1 FPGAs Architecture

FPGAs are composed of programmable interconnections connecting the Configurable Logic Blocks (CLBs) as shown in Fig.2.4 [28]. They also consist of programmable input/output (I/O) ports to allow communication with the surroundings via various peripherals [29]. In other words, the basic cells along with the interconnections are completely programable by the designer/end user. Thus, it allows building the required hardware architecture according to the application requirements [30]. Look Up Table (LUT), Multiplexers and Sequential elements are the three main components of CLBs. LUT is the main block in the FPGA. It is responsible for implementing the combinational logic by mapping it to the truth table via the SRAM configuration cells in the FPGA. Multiplexers are used to determine the signals connections in the CLB. As for the Sequential elements, they are important for the digital logic design. Sequential elements can be either latch or flip-flop and be edge or level sensitive.
2.4.2 Partial Reconfiguration

It should be noted that the logic in FPGAs are divided into two parts: static and reconfigurable presented in Fig. 2.4 as grey and black respectively. FPGAs can be considered as flexible devices since it can be programmed or re-programmed without the need of re-fabrication. This ability is known as Partial Re-configuration in which the modification takes place in the Internal Configuration Access Point (ICAP); while the FPGA is operating, a bit file known as partial configuration file is loaded without affecting the static part of the design [31]. As shown in Fig. 2.8, in order to modify the function implemented on Block “A”, one of the multiple bit files (A4.bit, A3.bit, A2.bit & A1.bit) available must be downloaded.
There are two forms of faults that take place in FPGAs: transient and permanent.

The complete damage of the semiconductors is known as the permanent fault. The reparation of this fault happens only when the faulty hardware is replaced or repaired. The main cause of this fault is due to either degradation phenomena, manufacturing defects or electromagnetic waves/radiation.

Negative-Bias Temperature Instability, Hot-carrier effect, Electro migration and Time-Dependent Dielectric Breakdown (TDDB) are the types of degradation phenomenon. Hot-carrier effect causes a stuck of charges in the gate channel interface region. This results in reducing the channel mobility gradually and increasing the threshold voltage in CMOS transistors [32]. As for the Negative-Bias Temperature Instability, the switching speed becomes slower causing delay faults and trapped charges as in the case of hot-carrier effect [33].

On the other hand, Electromigration is a process where the metal ions migrate over time consequently, voids and deposits in FPGA interconnects are created. This results in open and short circuits causing faults [34]. Time-Dependent Dielectric Breakdown (TDDB) degradation affects the transistors’ gates resulting in an increase in the leakage current which causes a short circuit. The mentioned mechanism causes charge traps within the gate [35]. Concerning the manufacturing defects, it is the type of fault where the circuit nodes are stuck at 0 or 1 or, switch too slow that it can’t meet the timing specification. FPGA interconnect network can be affected
by the defects, since it can result in short or open faults in the circuits and stuck open or closed pass transistors [36].

Electromagnetic waves/radiation can be considered the most common cause for inducing the fault. There are different types of faults due to the mentioned reason. Single Event Effects (SEE) occurs when the radiation hits the silicone of the FPGA causing either permanent or transient fault depending on the amount of charge [37].

Permanent faults result from SEE includes Single Event Latch up (SEL) which is an abnormal high-current state in the FPGA resulted from the flow of a single energetic particle through sensitive regions of the device structure and causing the loss of device functionality. SEL might cause permanent failure to the device. However, if the device is not permanently damaged, power cycling of the device is required to restore normal operation.

Transient faults are basically the ones where recovery is applicable by programming the correct bits. There are several types of transient faults resulted from SEE such as: Single Event Upsets (SEUs), Multiple Bit Upset (MBU), Single Event Functional Interrupt (SEFI), Single Event Transient (SET), Address decoding fault and Coupling fault. SEU is the most common one, where an SRAM cell in the configuration memory is being flipped from a ‘0’ to ‘1’ or vice versa, causing wrong functionality of the FPGA. The fault is corrected when the memory is re-configured with correct bits. The difference between SEU & MBU is that in the MBU, the fault occurs in several adjacent configuration bits. Regarding SEFI, the fault occurs when SEUs happens in the configuration RAM of the active region of an FPGA [38]. As for SET, this fault is due to induced voltage pulses on a combinatorial circuit in a device from the impacting ions. Invalid data values can be transmitted via the circuit, if the induced voltage level increases above the switching level threshold and the pulse-width is enough.

Concerning Address decoding fault, it is due to any fault affecting the address decoding causing address mismatches. On the other hand, coupling fault is the fault where a write operation to a cell changes the content of another cell.
2.5 Previous work on Fault tolerance on FPGAs

In [39], FPGAs were used and the fault tolerance and modular redundancy concepts were illustrated. In this study a fault secure FPGA-based TMR voter was developed. Since FPGA has lots of properties and advantages, a TMR voter employed on the FPGA is no longer taken as a single point of failure in the TMR modular redundancy solution. The TMR voter is executed on FPGA using Truth Table (TT) that applies alternating logic and is connected to a multiplexer. After that, the multiplexer is connected to a Flip-Flop (FF) which saves the voter’s output in order to be used by the rest of the system. It should be noted that, SEUs and SETs where used to represent transient faults while the stuck-at-0(1) represents permanent faults.

The TT data is controlled by the bits in the configuration memory of the FPGA. Hence, if a SEU occurs in the TT, it will directly affect the configuration memory and consequently, the TMR voter’s functionality will be affected. Thus, [39] proposes using the direct access of the ICAP to the configuration memory of the FPGA along with its ability to read and write data. Furthermore, it was shown that the wrong outputs produced from the TMR voter can be avoided by using alternating logic and the ICAP.

By masking the incorrect module’s output, the fault-secure TMR voter prevents the whole system from producing an erroneous output. However, it also indicates the faulty module’s address which caused a different output. This is attained by two additional output links namely A0 and A1 that carry the address of the faulty module, as shown in figure 2.9, as presented in [39].

![Figure 2.9 Fault Secure TMR voter Architecture](image)
In addition, to detect faults in the multiplexer and the FF, the technique of alternating logic is used. This method is used for fault discovery in a digital circuit and can be easily applied to self-dual functions. Self-dual functions satisfy the property of \( f(Y) = f'(Y') \) or converting any logical function, \( F \), into a self-dual one, \( F_{sd} \) by adding an extra input, \( x \), so that \( F_{sd} = xF + x'F_d \), where \( F_d \), is the dual of the original function \( F \). The application of alternating logic is done by sending the data at \( t_0 \) after that send its complement at \( t_0 + \Delta \). By this, a fault can be detected when the two sets of received data are not complements. [39] implemented this circuit and is shown in Fig. 2.10.

\[
\begin{align*}
D3 & \quad A4 \\
X & \quad A3 \\
M2 & \quad A2 \\
M3 & \quad A1
\end{align*}
\]

*Figure 2.10 Implementation of majority function voter on FPGAs [39]*

If a fault occurs in any of the voter components, alternating logic will be used for fault detection. Although, the recovery will be different, according to the type of fault that took place. If the fault is an SEU or MEU which occurs in the configuration memory of the FPGA, the ICAP will handle since it can read and write the voter’s configuration data. After reading the voter’s configuration data by the ICAP, it compares them with a hardwired previous copy that is suggested to be located inside the FPGA. Whenever a bit is found to be different, the ICAP overwrites the configuration bits using DPR. However, if a permanent fault occurs, the voter’s circuitry is reconfigured all over again, using DPR, on another segment on the FPGA [39].

In [40], a Fault-Tolerant FPGA-based S2A controller is modeled that consists of a small processor, memory and related hardware. Fault-Tolerant methods are implemented to the model. By this the single points of failure are eliminated even in recovery approaches. The method used for recovery is DPR. The fault model used is single and/or some multiple events upset. The system showed an enhanced reliability in the case study.

In [41], a new scheme is proposed that dynamically selects the suitable Fault Tolerance method which can be autonomously implemented to all system’s components. In this model,
there is a compromise between increasing reliability of the system and overriding the minimum amount of battery power according to the application requirements. Choosing the suitable FT technique depends on numerous changing factors throughout the mission time. These factors are: Battery level, module power consumption, failure type either SEU only, or SEU & MEU, module criticality at any time instant and module MTTF.

In [6], the FPGA-based controller (K) in the S2A model can be recovered from both transient and permanent faults. A spare location is added which is partially reconfigurable. This location is linked to the voter with the three duplicates of the controller (K). Whenever a permanent fault occurs in any of the modules, a new copy of the module is downloaded by the network to the spare location. Consequently, the system returns to TMR operation using two of the original system modules and the recently downloaded one in the partially reconfigurable location.

In [42], the FPGA is composed of: logic, interconnection and recovery part. Whenever a permanent fault occurs in a logic part, it is relocated in a recovery part. Triple Modular Redundancy (TMR) is as the FT techniques in FPGAs. If any fault occurs in one of the modules, other modules can cover the fault.

In Chapter 3, a detailed description of the model is presented along with the Riverbed simulations, results and performability analysis.
Chapter 3

Performability of Fault-Tolerant NCSs with Video Sensors

3.1 Proposed Model

The proposed model, as displayed in Fig.3.1, consists of 16 sensors/ sensing nodes, 4 actuators/ actuation nodes and their controller in addition to a video sensor with an embedded controller and video actuator. A 100B packet of control data is being sent by the 16 sensors; each sends every 694 µsec to the controller (K) and at the same time to the embedded video controller (KV). The reason for sending the exact data to both controllers is in case K fails it acts as a backup. UDP is the protocol used instead of TCP to dodge the congestion resulted from acknowledgments. After that, the controller transmits its produced control decision to the actuators after being processed. Simultaneously, every 1 sec the video sensor (SV) transmits the video data through the embedded video controller (KV) to the video actuator (AV). The model
has two watchdog timers; each sends 1B at half the sampling period to avoid network overloading and to ensure that each controller recognizes the changes before the processing of the coming packet. KV sends a watchdog signal to K every 0.5sec that is half the sampling period. Also, K exchanges a watchdog signal to the KV every 347μsec which is half the sampling period.

Since the proposed model is a real-time control system, it should abide by certain constraints in terms of delay and packet loss. In other words, system should guarantee that there are no packet drops and no over delayed packets. These constraints apply to the watchdog traffic along with the system’s normal control traffic from the sensor traffic, actuator traffic and controller traffic.

Regarding the control traffic, the overall end-to-end delay for packets transmitted from the sensors to the controller and from the controllers to the actuators should not exceed the system’s sampling period. On the other hand, the end-to-end delay for the watchdog traffic should not exceed half the sampling period [43]. The reason behind this is to allow adequate time for other controller to conquest in case any of the controllers fails. Zero packet loss should be guaranteed for both control and watchdog traffic.
In a factory, K is the controller that should be in charge of the drive of the conveyer belt, while SV is the camera sensor that measures the product; bottles for instance. Then it processes the video sensed data and respond accordingly through the embedded controller KV. The conveyer belt should always be in continuous movement. The system can stand the failure of either the two controllers without experiencing a problem in the production line. In the event where KV fails, some of the video data will be discarded; hence the system will safely fail by discarding the bottles. To elaborate, losing some video packets at the moment of KV failing before K taking over is not a problem. The other event is when K fails; the conveyer belt will not be affected since the sensors are already transmitting their data to KV. The scenarios were simulated using Riverbed Modeler and will be described in the following part.
3.2 Riverbed Simulations

Fig. 3.2 presents the Riverbed simulated model.

The three simulated parts are: Fault-free, Failure of the main controller K, and Failure of the embedded video controller KV.

3.2.1 Fault-free Scenario

The first part was an ordinarily operating scenario where all nodes are correctly functioning. This is when both controllers K & KV are operative. All the 16 sensors along with the video sensor send to both controllers, K & KV. At the same time, the watchdog application is being communicated between both controllers. Since the controllers are working therefore, they both collect data from all the sensing nodes, and undergo the required processing and after that transmit the needed action to the concerned actuation nodes.

The reason behind reviewing the normally operational environment is to ensure the correct procedure while the sensors are transmitting double the number of packets, since in this case there is an increased traffic sent over the network.
3.2.2 Fault-tolerant Scenarios

The second part is about simulating the failure of one controller while the other controller is taking over the full control load. Full load means that the remaining operative controller is responsible for receiving from sensing nodes, and replying to actuation nodes of both cells [17]. Accordingly, the functioning controller would do the required actions and respond to all the actuation nodes.

Now, the volume of traffic is much less than the traffic of the fault-free model since the replication is no longer there and the only congestion is in the connection between the functioning controller and its switch.

**Main controller (K) failure**

![Figure 3.4 K failure Noiseless Model](image)

In this case, the sixteen sensing nodes (S1 to S16) each send to the entrenched video controller (KV). At the same time, the video sensor (SV) transmits the data to KV. In other words, it transmits to itself since KV is entrenched in SV. After that, KV sends to the 4 actuators (A1 to A4) and to the Video actuator (AV). Thus, the raw and control data are not being sent on the network.
Video Embedded controller (KV) failure

With the failure of KV, the 16 Sensors (S1 to S16) each send its data to the main controller (K). At the same time, the Video Sensor (SV) transmits its data to K. Then, K transmits the processed action to AV and to A1, A2, A3 and A4. In this scenario, the raw data is communicated to the controller once KV fails. Consequently, both raw and control data are transmitted inside the network. The interarrival time is one second. Three cameras with different file sizes were tested. It was important to test various file sizes and track the behavior of the system accordingly.

According to the camera specifications, the raw data is being calculated by the below equation. Three different cameras were simulated [44].

\[
\text{File size (in KB)} = \frac{N \times M \times B}{8 \times 1024} \quad (5)
\]

where N and M are the bitmap dimensions and B is the color depth in bits.
For the 300 KB [45] camera

N equals to 640, M equals to 480 and B equals to 8

For the 768 KB [46] Camera

N equals to 1024, M equals to 768 and B equals to 8

For the 900 KB [47] Camera

N equals to 280, M equals to 20 and B equals to 8

3.2.3 Noise scenarios

Last, the third part included an additional noise node (N). N sends an FTP of 3800KB file size to K with a fixed interarrival time of 1 sec. This was simulated on all cases. N sends to K simultaneously while the system is normally operating. The fault free scenario, where the 16 controllers are sending their data to K as well and to KV as a backup. The embedded video control is also sending its required action to the video actuator. Along with two watchdogs’ applications are being exchanged. It should be noted that the FTP in this case signifies a constant form of communication. The FTP is TCP/IP which is a constant data exchange method. It is considered a noise factor over the system, which is using UDP application to simulate the packets sent and received from sensors to controllers and from controllers to actuators.
Figure 3.6 fault free Noisy Model
Figure 3.7 failure Noisy model
3.3 Simulation Results

The system performance is evaluated by measuring the end-to-end delay. This end-to-end delay should cover all delay types: encapsulation, processing, propagation, queuing and decapsulation.

Table 3.1 shows the maximum end-to-end delay from sensing node to controller as well as from controller to actuation node for all scenarios.

Table 3.2 presents the maximum end-to-end delay from sensing node to controller as well as from controller to actuation nodes for all scenarios, but with the addition of the Noise node.

As for Fig. 3.8, it displays the End-to-End delay of K in the case of fault-free noiseless Scenario.
From the below mentioned results, the main controller failure (KF) has the minimum delay. This is expected since the sensing nodes transmit their data only to the embedded video controller. While the raw data is not transmitted on the network it is still between the video sensor and its embedded controller. On the other hand, the embedded controller failure scenarios showed different results. Regarding the 300 KB camera, it shows a delay that is less than the fault free scenario. To illustrate, the sensing nodes in this case transmits the data and the raw video data to the main controller only, the amount of traffic results in a delay that is still less than Fault Free (FF). Unlike, the raw video data sent by the 768 KB camera and 900 KB camera which results in network congestion. This congestion causes an increase in the end-to-end delay to be greater than the FF scenario.

It should be noted that the collision domains are isolated, since the network is switched. Thus, periodic traffic will cause deterministic end-to-end delays.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Delay µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>12.72</td>
</tr>
<tr>
<td>KF</td>
<td>10.58</td>
</tr>
<tr>
<td>KVF_300 camera</td>
<td>11.38</td>
</tr>
<tr>
<td>KVF_768 camera</td>
<td>14.14</td>
</tr>
<tr>
<td>KVF_900 camera</td>
<td>15.34</td>
</tr>
</tbody>
</table>

Table 3.1 End to End delay for noiseless Scenarios
Table 3.2 End to End delay for noise Scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Delay µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>614.94</td>
</tr>
<tr>
<td>KF</td>
<td>600.56</td>
</tr>
<tr>
<td>KVF_300 camera</td>
<td>601.37</td>
</tr>
<tr>
<td>KVF_768 camera</td>
<td>604.04</td>
</tr>
<tr>
<td>KVF_900 camera</td>
<td>605.26</td>
</tr>
</tbody>
</table>

Figure 3.9 Packet End-to-End Delay of K in FF noiseless Scenario
3.4 Performability Analysis

As explained above, the system can stand one of the two controllers failing, with no observable decrease in the system’s performance. As in [25], the number of operational controllers affects the end-to-end delay.

In order to compute the Transient Performability TP (t), the system reliability model should be expressed first. The Markov model for the system is shown in the below Fig.3.9, it has 4 states: 3, 2, 1 and 0. State 3 represents the fault-free while state 2 is when K fails, and state 1 is when KV fails. State 0 indicates the system failure.

Assume $\lambda_K$ to be the failure rate of the main controller while $\lambda_{KV}$ is the failure rate of KV. In this model for simplicity, it is supposed that, for all three cameras, the controllers have the same failure rate $\lambda_{KV}$.

After that, a reward (or a penalty) should be given for each state [48], [18]. In this case, the reward is defined to be the difference between the sampling period or in other words packet delivery deadline that is 694μs and the end-to-end delay.
The Chapman-Kolmogorov equations shown below are used to calculate the transient probability of being in any of the 4 states [20].

Let $P_i(t)$ be the probability of being in any of state $i (i = 3, 2, 1, 0)$ at time $t$. Given that $P_3(0) = 1$ and $P_2(0) = P_1(0) = P_0(0) = 0$

\[
\frac{dP}{dt} = P \times T
\]  \hspace{1cm} (6)

\[
P = \begin{bmatrix} P_3 & P_2 & P_1 & P_0 \end{bmatrix}
\]  \hspace{1cm} (7)

\[
T = \begin{bmatrix} -\lambda_k - \lambda_{kv} & \lambda_k & \lambda_{kv} & 0 \\ 0 & -\lambda_{kv} & 0 & \lambda_{kv} \\ 0 & 0 & -\lambda_k & \lambda_k \\ 0 & 0 & 0 & 0 \end{bmatrix}
\]  \hspace{1cm} (8)

Substituting $P$ & $T$ in $\frac{dP}{dt}$

\[
\frac{dP_3}{dt} = P_3(-\lambda_k - \lambda_{kv})
\]  \hspace{1cm} (9)
Taking \( \ln \) to both sides

\[
\ln(P_3) = (-\lambda_k - \lambda_{KV})t
\]

(10)

Thus,

\[
P_3(t) = e^{-\lambda_k - \lambda_{KV}t}
\]

(11)

Substituting \( P \) & \( T \) in \( \frac{dP}{dt} \)

\[
\frac{dP_2}{dt} = \lambda_k P_3 - \lambda_{KV} P_2
\]

(12)

\[
\frac{dP_2}{dt} + \lambda_{KV} P_2 = \lambda_k P_3
\]

(13)

\[
\frac{dP_2}{dt} + \lambda_{KV} P_2 = \lambda_k (e^{-\lambda_k t})
\]

(14)

Integrate both sides

\[
P_2 e^{\lambda_{KV}t} = \int \lambda_k e^{-(\lambda_k + \lambda_{KV})t} e^{\lambda_{KV}t} dt + C
\]

(15)

\[
P_2 e^{\lambda_{KV}t} = \int \lambda_k e^{-(\lambda_k + \lambda_{KV})t} e^{\lambda_{KV}t} + C
\]

(16)

\[
P_2 e^{\lambda_{KV}t} = \int \lambda_k e^{-(\lambda_k t)} + C
\]

(17)

\[
P_2 e^{\lambda_{KV}t} = -e^{-(\lambda_k t)} + C
\]

(18)

\[
P_2 = -e^{-(\lambda_{KV} + \lambda_k) t} + C
\]

(19)

Let

\[
P_2(0) = 0
\]

(20)
Therefore,

\[ C = 1 \]

Thus,

\[ P_2(t) = e^{-\lambda_K t} - e^{(-\lambda_k-\lambda_K)t} \quad (21) \]

Substituting P & T in \(\frac{dP}{dt}\)

\[ \frac{dP_1}{dt} = \lambda_K P_3 - \lambda_K P_1 \quad (22) \]

Integrate both sides

\[ P_1 e^{\lambda_k t} = \int \lambda_K e^{(-\lambda_k-\lambda_K)t} e^{\lambda_k t} + C_2 \quad (23) \]

\[ P_1 e^{\lambda_k t} = \int \lambda_K e^{(-\lambda_K)t} + C_2 \quad (24) \]

\[ P_1 = e^{(-\lambda_k-\lambda_K)t} + \frac{C_2}{e^{(\lambda_k)t}} \quad (25) \]

Let

\[ P_1(0) = 0 \quad (26) \]

Then

\[ C_2 = 1 \quad (27) \]

Therefore

\[ P_1(t) = e^{-\lambda_K t} - e^{(-\lambda_k-\lambda_K)t} \quad (28) \]

\[ P_3 + P_2 + P_1 + P_0 = 1 \quad (29) \]

The Transient Performability TP (t) is calculated as follows [15]:

39
\[ TP(t) = \sum_{i \in \psi} P_i Rew_i \]  

(30)

Where \( \psi \) is the set of the three states in the mentioned model and \( Rew_i \) is the reward of state \( i \).

Table 3.3 shows the Reward which is the difference between the 694\( \mu s \) sampling period and the end-to-end delay as mentioned before and the same delay values presented in Table 3.3.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Delay ( \mu s )</th>
<th>Reward ( \mu s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>12.72</td>
<td>681.28</td>
</tr>
<tr>
<td>KF</td>
<td>10.58</td>
<td>683.42</td>
</tr>
<tr>
<td>KVF_300 camera</td>
<td>11.38</td>
<td>682.62</td>
</tr>
<tr>
<td>KVF_768 camera</td>
<td>14.14</td>
<td>679.86</td>
</tr>
<tr>
<td>KVF_900 camera</td>
<td>15.34</td>
<td>678.66</td>
</tr>
</tbody>
</table>

In this case study, it is supposed that the entrenched controller fails 2 times per month while the main controller fails 1 time per month.

\[ \lambda_{kv} = \frac{2}{24 \times 30} = 0.002778 \text{ per hour} \]  

(31)
\[ \lambda_k = \frac{1}{24 \times 30} = 0.001389 \text{ per hour} \quad (32) \]

Fig. 3.11 displays the Transient Performability for the three cameras: 300KB, 768KB and 900KB over 16 hours when KV fails.

![Performability graph showing TP_300, TP_768, TP_900 over 16 hours]

**Figure 3.11** Performability for the 300KB, 768 & 900KBs camera for 16 hours.

### 3.5 Summary

Smart sensing nodes in NCS are growing in importance with the emerge of Industry 4.0. Utilizing smart sensing nodes can propose lots of benefits to the ordinary NCS. Besides, the use of these smart sensing nodes can be extra achievable specifically due to the use of NCSs that offer plenty of bandwidth. Since Industry 4.0 is concerned about renovating ordinary factories to smart and Network Control System is composed of smart sensing nodes, actuation nodes and controllers. Thus, the use of NCS in automation applications should be of great importance in the upcoming industrial revolution, Industry 4.0.
The proposed model is categorized as NCS that can be used in Industry 4.0. The above-mentioned model presents an NCS that utilizes switched Ethernet, with smart sensors, smart actuators and a smart controller. The system includes a smart sensor that processes video data through an onboard camera and embedded video controller. This chapter studied the effect of the video sensor on the NCS and presented a fault-tolerant system that shows the use of the sensor’s onboard controller as a backup to the system’s main controller.

All sensing nodes send their samples to all the controllers via the ethernet switch that is wired to the controller. On the controllers’ level, Fault tolerance is implemented, and the system is shown to stand the failure of one of the controllers without a performance degradation that, and the system is shown to tolerate the failure of either controllers without a performance degradation that causes violations of packet delivery deadlines. In the incident of one controller failing, the other functional controller carries the load. The network is shown to tolerate the failure of one controller as long as the other one is functioning. The remaining functional controller handles traffic during the failure of the other one. Both controllers must be aware of the real-time status of each other, in order to guarantee a smooth transition in case of failure. Both controllers receive the sampled data sent by the sensors simultaneously. The two controllers exchange a watchdog application to update each other of their status. Hence, at any time, the two controllers are fully aware of the status of the network. When the embedded video controller fails, the raw video data is communicated on the network. Three different cameras with different features were tested.

The model is successfully modeled and tested using Riverbed Network Modeler. Lastly, a performability analysis is presented on the described model.

A performability analysis of the system was then conducted in which a case study that simultaneously examines failure data with reward for all of the simulated scenarios was examined.

In the coming chapter, a modification to the presented system is introduced where an FPGA is added to the system, to make use of its powerfulness and increase the system’s reliability. In addition, a new important feature is added to the system, storage in which video images are captured and stored.
Chapter 4

Reliable FPGA-based Camera Sensor for NCS

4.1 Modified model

In the previously explained chapter, the model is composed of 16 sensors, 4 actuators, a controller. In addition, a camera sensor with an entrenched video controller and its related video actuation node. The control data is being sent from the 16 sensors to the controller and to the embedded video controller as a backup in case the controller fails. After that, the controller then transmits its action to the actuation node. Concurrently, the video data is being transmitted from the camera sensor to its video actuator via the embedded video controller. Two watchdog signals were added for Fault Tolerance, the signals are sent at double the sampling rate which is half the sampling period to guarantee the functionality of the system when one controller fails. Thus, once any of the controllers fail, the functional controller can conquer its duties before processing the following packet. Three different scenarios were studied and simulated in the presence and absence of the noise node using Riverbed Network Modeler.

The difference between the previous model and the suggested one is the installation of an FPGA chip instead of the video controller. The model still has 16 sensors S1=>S16, a controller K and 4 actuators A1=> A4. As shown in the below figure, the FPGA consists of two similar soft-core camera controllers namely: KV1 & KV2. In addition, KV3; it is another soft-core camera controller that is used for storage. The main task of KV3 is to locally save the video data. Every now and then, KV3 sends the saved data to the supervisor over the network. There are also two hardwired ARM processors in the FPGA as in [49]. In this model, the two ARM processors are considered as one powerful controller, K\text{ARM}. In error-free scenario, both the sensors send their data to K and K\text{ARM}. K\text{ARM} backs up K and takes over its responsibilities upon its failure. Since K\text{ARM} is powerful, it takes over the responsibilities of KV3 if it fails. Therefore, the system has three watchdogs. First, is between KV1 and KV2 where watchdog packet is being exchanged between both controllers in order to backup each other upon the failure of any of them. Second, KV3 transmits a watchdog packet to K\text{ARM}. Once KV3 fails, K\text{ARM} will notice this failure and
take over the duties of KV3. The last watchdog is between K\textsubscript{ARM} & K where K\textsubscript{ARM} can immediately handle K’s tasks in case of failure since it already has the data.

![Diagram of FPGA Controllers](image)

**Figure 4.1** FPGA Controllers

Unlike the previous model, the raw video data is not sent over the network. This is because either KV1 or KV2 will process the data in the FPGA; by this, the volume of data transferred on the network is less than in the faulty scenario mentioned in chapter 3 when the camera controller fails and the system will send the raw video data over the network to K. In the new architecture, the error-free and erroneous scenario are similar to the error-free scenario in chapter 3, this from the network point of view. Thus, this architecture will also meet control system restrictions in terms of the end-to-end delay and zero packet loss.

This model can be used in the same application previously suggested in the above chapter, namely an inspection machine. Though, in this case the video data is saved and transmitted to the supervisor from time to time.

Industrial environments are susceptible to several radiation sources. These sources are considered the primary source of errors in SRAM-based FPGAs. As previously explained, Single Event Effects (SEEs) resulted from radiations caused the existence of transient and permanent faults according to the amount of energy transferred by the radiation particles \[37\]. Unlike K\textsubscript{ARM},
KV1, KV2 and KV3 are subjected to both impermanent and permanent failures since they are FPGA-based. KARM is mainly affected by permanent failures. Single Event Upsets (SEUs) is well known as the most common type of failure in FPGAs [37]. Permanent failures which destroy the fabric of the FPGA can occur, depending on the amount energy hitting the FPGA. On the other hand, impermanent failures can be recovered by using DPR in order to reconfigure the FPGA with the correct bit streams [50].

### 4.2 Reliability Modeling

Fig. 4.6 presents the Markov model for the suggested system. In this model, the sequence of modules is arranged as follows: KV1, K, KARM, KV2 & KV3 respectively. To illustrate, in the Markov model, a “0” indicates that the module is functioning while a “1” signifies that it has failed. Therefore, “00000” is the initial state where all the elements are operating. In case of fault in either K and KARM, the system goes to states “01000” and “00100” respectively since, they are only affected by permanent failures. Hk and Harm are the notation for the transition rates of K and KARM respectively. As for the failures of KV1, KV2 and KV3, the case is dissimilar. As, these 3 controllers are prone to both impermanent as well as permanent failures. An attempt for recovery is operated and Dynamic Partial Reconfiguration (DPR) is initiated once a failure is sensed in any of these controllers [50]. If the failure is impermanent, the error will be successfully repaired by DPR. Though, if the module still does not operate properly after DPR, the failure should be permanent, and the controller is removed from the system.

![Figure 4.2 An example of system behavior in the case of KV1 failure](image)

Figure 4.2 An example of system behavior in the case of KV1 failure

45
In the presented Markov model shown in the below Fig. 4.6, it is assumed that KV1, KV2 and KV3 have equal failure rates, since the three are all implemented with similar soft-core processors such as the MicroBlaze. For the given model, assume $S$ to be the rate of impermanent failures affecting KV1, KV2, or KV3 and $H$ the rate of permanent failures for KV1, KV2, or KV3. Fig. 4.2 illustrates a section of the Markov model in Fig.4.6 to explain the system behavior in the case of impermanent or permanent failure of KV1, KV2 or KV3. The figure shows the transition from the error-free state 00000 to KV1 failure state 10000. Upon an error, the system moves to state “rec” at a rate of $S+H$. At “rec” state, DPR is performed on the location of KV1 in the FPGA. Now comes the decision, if the error is impermanent, KV1 continues its correct operation and the system transitions back to the error free state 00000 at a rate of $Z\times \mu$. On the other hand, if the failure is permanent, the system transitions to the error state 10000 at a rate of $(1-Z) \times \mu$. It should be noted that the mean time needed to complete DPR in the location of KV1 is $(1/\mu)$ and “$Z$” is the conditional probability in which a failure is impermanent given that a failure has happened. The same approach is used with the rest of the model.
According to the previously explained approach the transition in the Markov model goes as follows. Starting from state “00000”, a failure can occur in any of the 5 controllers. If the failure happens at KV2, then the next state becomes “rec1” at which the type of failure is determined whether temporary or permanent. If the failure in KV2 is temporary, then the node recovers and goes back to the previous state, in this case the starting state “00000”. Else if the failure is permanent, then KV2 is now out of service and noted as “1”. Back to the initial state “00000”, if a failure occurs in KV3, the next state becomes “rec” in which also the type of failure is determined and the node status is set accordingly to “0” upon recovery from temporary failure or “1” if it’s a permanent failure. The case where KV1 has failed was previously explained and presented in Fig. 4.2 transitioning from “00000” to “01000” indicates the failure of the main
controller K while moving to state “00100” means that a failure has occurred in K_{ARM}. Fig.4.3 summarizes the explained transitions.

![Diagram](image)

**Figure 4.4** Transitions from state “10000”

Now, moving to the second stage in the Markov model where an error has occurred in one of the controllers. To illustrate, there are four probabilities when transitioning from state “10000” where KV1 has permanently failed. First, the failure of K indicated as “11000”; in this case the system is still functioning since K_{ARM} will take over K’s function and KV2 is already replacing KV1. Second, the failure of K_{ARM} and being in state “10100” where the system is correctly working, but with K taking over K_{ARM} and KV2 taking over KV1. Third, the failure of KV3 and then moving to “rec9” in order to identify the type of failure. As previously explained, if the failure is permanent, then K_{ARM} will take over KV3’s function and the system will still work properly. On the other hand, if the failure is temporary, the next stage will be back to “10000”. Last but not least, if KV2 permanently fails going to “10010”, then the system stops since this is one of the down states where both video controllers have failed. This explained concept for stage 2 to apply to all states “01000”, “00100”, “00010” and “00001”.

48
Figure 4.5 Transitions from state "01010"

As for stage 3, where two controllers have failed, there are three probabilities for each state. For example, there are three possible scenarios when moving from stage “01010” where both K and KV2 have failed. To start with, the failure of KV3 and being in the recovery state to check the type of failure. If KV3 permanently fails, the next state is “01011” and the system will still work properly, as KARM will take over. On the other hand, once KARM fails the system will move to one of the down states “01110” and will stop working. The same thing happens upon the failure of KV1, the next state will be “11010” which is also one of the down states. The same approach is taken in the rest of stage 3 states.

Moving to the last stage in the Markov model where three controllers have failed, but the system is still functioning as in states “01011” and “11001”. This because in state”01011”, KV1 is taking over KV2 and KARM is taking over both KV3 & K. Same concept applies to “11001”, where KV2is taking over KV1 and KARM is taking over both KV3 & K.
The system reliability can be obtained by solving the Markov model. The system has 58 states. Let \( P_i(t) \) be the probability of being in state \( i \) at time \( t \). Using the Chapman-Kolmogorov equations, the transient probability of residing in any of the \( i \) states can be calculated [20]:

\[
\frac{dP}{dt} = P \times T \quad (33)
\]

where

\[
\frac{dP}{dt} = \frac{dP_{00000}}{dt} \ldots \ldots \frac{dP_{11011}}{dt} \quad (34)
\]

\[
P = (P_{00000} P_{10000} \ldots \ldots P_{01111} P_{11011}) \quad (35)
\]

\[
T = \begin{bmatrix}
- (3S + 3H + Hk + Harm) & \ldots & 0 \\
Z \times \mu & \ldots & 0 \\
\ldots & \ldots & \ldots \\
0 & \ldots & 0
\end{bmatrix} \quad (36)
\]

The probabilities of each state \( i \) is included in the matrix \( P \) while \( T \) is the Transition Rate Matrix. Assume the probability of \( P_{00000}(0) \) is 1 while all other states have an initial probability of 0. By substituting \( T \) and \( P \) in equation (1), the Chapman-Kolmogorov equations can be solved. Besides, \( P_i(t) \) for each state \( i \) can be calculated. Consequently, system reliability \( R(t) \) at time \( t \) can be calculated by summing the probabilities of not being in any of the “down” states at time \( t \). It should be noted that this is where there is a complete system failure. When both KV1 and KV2 fail or both KARM and KV3 fail or both K and KARM fail, the system should fail. The below table 4.1 shows the down states.
Table 4.1 Down States

<table>
<thead>
<tr>
<th>Down States</th>
<th>Failed controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010</td>
<td>KV1 &amp; KV2</td>
</tr>
<tr>
<td>00101</td>
<td>KARM &amp; KV3</td>
</tr>
<tr>
<td>01100</td>
<td>K &amp; KARM</td>
</tr>
<tr>
<td>00111</td>
<td>KARM, KV2 &amp; KV3</td>
</tr>
<tr>
<td>01110</td>
<td>K, KARM &amp; KV2</td>
</tr>
<tr>
<td>01101</td>
<td>K, KARM &amp; KV3</td>
</tr>
<tr>
<td>11010</td>
<td>KV1, K &amp; KV2</td>
</tr>
<tr>
<td>11100</td>
<td>KV1, K &amp; KARM</td>
</tr>
<tr>
<td>10101</td>
<td>KV1, KARM &amp; KV3</td>
</tr>
<tr>
<td>10011</td>
<td>KV1, KV2 &amp; KV3</td>
</tr>
<tr>
<td>11011</td>
<td>KV1, K, KV2 &amp; KV3</td>
</tr>
<tr>
<td>01111</td>
<td>K, KARM, KV2 &amp; KV3</td>
</tr>
<tr>
<td>10110</td>
<td>KV1, KARM &amp; KV2</td>
</tr>
</tbody>
</table>

Let $P_F(t)$ be the probability of being in any of the down states. Hence,

$$R(t) = 1 - P_F(t) \quad (37)$$
Figure 4.6 Markov Model on SHARPE
4.3 Case Study

In this section, a comparison case study between the recently presented system and the previous system is presented. An answer for which system gives better reliability is given.

It is important to note that one of the advantages of the proposed system is the storage function where KV3 saves the images and sends them to the supervisor every now and then. In addition to the mentioned advantage, the raw data is NEVER sent on the network once the video controller fails unlike the previous system. The reason for this is that the two video controllers, KV1 & KV2, are considered a 1-out-of-2 fault-tolerant system in the FPGA.

The previous system was a simple 1-out-of-2 system where the embedded video controller would back up the main controller upon its failure and vice versa. For reasonable comparison, the same failure rates used in section 3.4 is used in this case study.

The failure rate for the main controller was 1 per month; thus, \( H_k \) is equal to 1 per month that is 0.25 per week. As for the video controller, the failure rate in the previous system was 2 per month. Similarly, the failure rate of the FPGA will be 2 per month that is 0.5 per week. It should be highlighted that these failure rates consider the harsh industrial environment.

According to the layout of the Xilinx Zynq device in the Xilinx Vivado tool, it is shown that the area consumed by the two ARM processors is less than the area of the FPGA. As a result, assume the following:

\[ H_{arm} = 0.3 \times 0.5 = 0.15 \text{ /week} \quad (38) \]

As previously mentioned, KV1, KV2 & KV3 are implemented by similar softcore MicroBlaze. Reference to [52], the Single Event Upsets (SEUs) per bit rate was \( 2.17 \times 10^{-9} \) /week. Supposed that the size of the MicroBlaze processor is 2036 KB, \( S=0.036/\text{week} \). This rate applies to the three softcore processors. Reference to the information in [53] for the 3 softcore MicroBlaze processors assume that failure rate \( H \) is 250 times higher than the rate of SEUs which is equivalent to 0.000144/week.
\[ \lambda_H = 250 \times \lambda_S \]  

(39)

As for the repair rate \( \mu \), it depends on the time taken to perform DPR [50] on a MicroBlaze processor. In this case and following the information in [38], the repair rate \( \mu \) is equal to the following:

\[ \mu = \frac{1}{MTTR} \]  

(40)

\[ \mu = 144034/\text{week} \]

The below Table 4.2 shows a comparison between the reliabilities of the proposed system with the FPGA and the previous system. In addition, it compares the mentioned systems to the non-fault tolerant scenario of the previously mentioned model in Chapter 3 namely, \( R_{\text{simplex}} \). The results were obtained using SHARPE package [51]. As shown in the results, the reliability of the presented system is better than the previous one although the existence of the new function, storage.

<table>
<thead>
<tr>
<th>Time (weeks)</th>
<th>( R_{\text{modified}} ) (%)</th>
<th>( R_{\text{previous}} ) (%)</th>
<th>( R_{\text{simplex}} ) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>96.914995</td>
<td>91.296489</td>
<td>47.23665527</td>
</tr>
<tr>
<td>2</td>
<td>89.790849</td>
<td>75.127994</td>
<td>22.31301601</td>
</tr>
<tr>
<td>3</td>
<td>80.862007</td>
<td>59.009749</td>
<td>10.53992246</td>
</tr>
<tr>
<td>4</td>
<td>71.456369</td>
<td>45.342766</td>
<td>4.978706837</td>
</tr>
<tr>
<td>5</td>
<td>62.327648</td>
<td>34.507205</td>
<td>2.351774586</td>
</tr>
<tr>
<td>6</td>
<td>53.871324</td>
<td>26.180823</td>
<td>1.110899654</td>
</tr>
</tbody>
</table>
4.4 Summary

Networked Control Systems (NCSs) is considered a motivating architecture for industrial applications. NCSs are composed of three main types of nodes: smart sensors, actuators and controllers. In addition, FPGAs have recently been increasingly used in industry. Dynamic Partial Reconfiguration (DPR) is an important feature in FPGAs that is one of the main advantages to use it, as it repairs some failures without disturbing system operation. The proposed modified model joint both NCSs and FPGAs in a new architecture that has numerous compensations. First, video saving ability is included that is crucial in some applications. Second, the model has various fault-tolerant features to extend system lifetime. Third, the raw video data is not transmitted over the network even in the case of failure. At last, a case study is presented to asses and measures the increase in reliability of the proposed model. Markov models were established and using SHARPE package the reliability of the presented system is compared with the system presented in chapter 3; the results show that the presented model is more reliable.

The modified proposed system used FPGAs since; they are one of the most powerful chips for high performance in industrial systems. It combines both NCS system with an FPGA instead of the Video Controller proposed in the previous model. The new system added storage feature where the images now can be stored. From time to time the stored data can be sent to the supervisor. Also, the stored data can be retrieved whenever it's essential. The use of FPGAs decreased the amount of traffic on the network once the video controller fails when compared to the previous model. Besides, there are two video controllers in the FPGA that backup each other in case of faults.

For reasonable Reliability analysis and fair comparison Markov models were presented & studied for both systems. This was simulated through SHARPE package to obtain system reliability and it was concluded that the proposed system showed improved reliability results although the proposed system had an added feature, storage. An example to demonstrate the results, after the 3rd week of operation, it was shown that the reliability of the previous system was 45% while the proposed system reliability was 71%.

In chapter 5, the thesis is concluded and some recommended future work will be discussed
Chapter 5

Conclusions and Recommended Future Work

Nowadays, Networked Control Systems play an important role in different fields and lots of industrial applications. The use of smart nodes: actuators, sensors & controllers offers lots of advantages in comparison to the traditional networks. At the same time, Industry 4.0 is concerned with transforming traditional factories to smart factories. Hence, deploying smart NCS in factories is part of Industry 4.0. Using FPGAs in NCS is also beneficial and important. Since they are powerful integrated circuits that can be used to apply any digital circuit to serve various applications in different fields. One of these fields is the industrial control. FPGAs are considered as protection of investment since updates or changes can be easily programmed and configured without the need of doing major changes in the system.

In this research, NCS that is composed of 16 sensors, controller and 4 actuators along with a video sensor with an embedded controller and their video actuator are modeled using Riverbed. Fault tolerance was added to the model at the controller level. Two watchdog applications are exchanged between the two controllers, so once any of the controller fails, the other takes over since it acts like a backup to the node. The system proved that it can tolerate the failure of one of the controllers. The simulations showed that the model was able to meet the system’s requirements with zero packet loss. Several scenarios were simulated. First, was the fault free scenario where all nodes are working correctly. After that a faulty scenario was tested were the main controller has failed. Then, the scenario where the embedded video controller has failed was simulated, with three different values for the camera: 300, 768 & 900 KB respectively. All the mentioned scenarios were tested in both the presence and absence of noise. As expected, the main controller’s failure scenario resulted in the least delay, since the raw video data isn’t sent on the network as in the failure of the embedded controller and the data isn’t duplicated when compared to the fault free scenario. The 300 KB video data when the embedded video controller failure resulted in a less delay in comparison to the FF. This is because the transmitted data is less than that of the FF unlike the 768 & 900 KB scenarios which congest the network. Last but not least, a performability analysis was then presented and a case study is
studied that concurrently inspects failure data with reward for all simulation scenarios. The modeled network uses switched Ethernet.

A modification for the presented model was then studied in which an FPGA was used instead of the camera controller. The FPGA consists of two hardwired ARM processors that acts as one powerful processor, K_{ARM}, and two similar soft-core camera controllers, KV1 & KV2, besides, an extra soft-core controller, KV3 that is used for storage. The new model provides two important features. First, storage which was absent in the previous model and is important in some applications. The other feature is that the raw data is never transferred on the network unlike the previous model. An important advantage is that the KV1 and KV2 back up with an identical controller. Also, K_{ARM} backs up KV3 and vice versa. Besides, the main controller K and K_{ARM} back up each other.

For fair comparison, a reliability analysis for both systems was presented. Markov models were developed and simulated using SHARPE. The results showed better reliability results for the modified system in comparison to the other one even though the modified had an additional function. For instance, by the third week of operation, the reliability of the previous system decreased to 45% unlike the reliability of proposed system that was still 71%.

In conclusion, the models presented in the research can be utilized in several applications one of which is the inspection machine. The main controller would be responsible for the movement of the conveyer belt while the video sensor would sense the item/product that should be inspected, process the video data and take action accordingly through the embedded controller. On the other hand, in the modified model, the video data is being stored and sent to the supervisor every now and then for reference. The conveyer belt should always be working. However, the system can tolerate controllers failing, without completely stopping the production line. The system also guarantees the fail secure.

The recommended future work to this research would be the following. Modeling the modified system on an FPGA and testing its performance, hence making a prototype for the model. Another suggestion would be using higher video quality and monitoring the delay and system’s requirements. Expanding the network by adding another identical cell to the introduced system and connect it with the previous one, by this the number of nodes would double, different
fault tolerance techniques at the controllers. After this, reliability analysis can be studied and the performance can be evaluated accordingly.
References

[1] Industry 4.0, the Smart Factory & the Digital Supply Chain,
    https://www.datexcorp.com/industry-4-0-smart-factory-digitization-supply-chain/ , last accessed 15/10/2018


Official site for SHARPE: http://sharpe.pratt.duke.edu

List of Publications from the Thesis

This work was published as follows:
