Communication Protocols, Queuing and Scheduling Delay Analysis in CANDU SCWR Hydrogen Co-generation Model

By

Fayyaz Ahmed

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Applied Science

in

The Faculty of Engineering and Applied Science

Electrical and Computer Engineering

University of Ontario Institute of Technology

August 2011

ABSTRACT

Industrial dynamical, Networked Control Systems (NCSs) are controlled over a communication network. We study a continuous-time CANada Deuterium Uranium-Super Critical Water Reactor **(**CANDU-SCWR) hydrogen plant and a discrete-time controller, sensor and actuator block, that are connected via a communication network, such as e.g. controller area network (CAN), Ethernet or wireless networks. Issues associated with NCSs are time-varying delays, time-varying sampling intervals and loss of data due to packet drop outs. Delays are also associated with software chosen, control system architecture and computation load. CANDU-SCWR hydrogen co-generation model reliability can be analyzed by dynamic flow graph methodology. We have analyzed the CANDU-SCWR feed water integration with the oxygen unit of copper chloride cycle and also conducted an analytical review of the current networked control system delays.

Keywords: CANDU SCWR hydrogen co-generation model, communication delays, queuing delays, scheduling delays.

TABLE OF CONTENTS

NOMENCLATURE

Ada	Structural Programming Language
AIFS	Arbitration Inter Frame Space
ALD	Application Layer Delay
AMD	Advanced Micro Devices
AODV	Adhoc On Demand Distance Vector
AP	Access Point
API	Application Programming Interface
BD	Back off Timer Delay
BEB	Binary Exponential Back off Algorithm
BSS	Basic Service Set
BSA	Basic Service Area
BWR	Boiling Water Reactor
C	General Purpose Programming Language
CA	Collision Avoidance
CAN	Controller Area Network
CANDU	Canada Deuterium Uranium
CD	Collision Detection
CD_e	Communication Delay
CDF	Complementary Cumulative Distribution Function
CFP	Contention Free Period
CMD	Communication Medium Delay
CP	Contention Period

CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/ Collision Detection
CFP	Contention Free Period
CW	Contention Window
CWmax	Contention Window Maximum
CWmin	Contention Window Minimum
CSRD	Cyclic Send & Request Data with Acknowledge
DA	Destination Address
DEL	Total Delay
DFM	Dynamic Flow Graph Methodology
DLC	Data link control
DCF	Distributed Coordination Function
DIFS	DCF Inter frame space
DFM	Dynamic Flow Graph Methodology
DS	Distributed System
Ds	Device Status
DSSS	Direct Sequence Spread Spectrum
DS_z	Data Size
ED	End of Delimiter
EOF	End of Frame
FC	Frame Control

FCS	Frame Checksum
FDL	Field bus Data Link
FHSS	Frequency Hopping Spread Spectrum
GTS	Guaranteed Time Slots
HCD	Hardware Computational Delay
HD	Hardware Delay
HP	High Pressure Turbine
HRD	Hardware Response Delay
Hx (X)	Heat Exchanger
IP	Intermediate Pressure Turbine
IR	Infra Red
IRQ	Interrupt Requests
ISM	Industrial, Science, Medical
IFS	Inter frame space
LE/ LEr	Length byte for Data
LP	Low Pressure Turbine
LWR	Light Water Reactor
MAC	Medium Access Control
Mac	MAC Header
MANET	Mobile Adhoc Networks
MAP	Manufacturing Area Protocol
M/G/1	Memory less, General, Single Server
MIPS	Microprocessor without Interlocked Pipeline Stages

MLD	MAC Layer Delay
M/M/ 1	Markov Memory less Single Server
MPa	Mega Pascal
MVL	Multi Value Logic
MWe	Mega Watt Electrical
MWM	Maximum Weight Matching
NAV	Network Allocation Vector
Nb	Number of Bytes
NC	Network Condition
NCS	Networked Control System
NRC	US National Regulatory Commission
OLD	OSI Layer Delay
OSI	Model Open Systems Interconnection Model
PCF	Point Coordination Function
PF	Protocol Frame Delay
FD	Frame Delay
PHY mode	Physical Layer mode, coding and modulation scheme
Phy	PHY Header
P&ID	Piping and Instrumentation Diagram
PLCP	Physical Layer Convergence Protocol
PLD	Physical Layer Delay
PIFS	PCF Inter Frame Space
PIM	Parallel Iterative Matching

POSIX	Portable Operating System Interface
POST	Post Processing
PowerPC	Performance Optimization with Enhanced RISC – Performance Computing
PPM	Pulse Position Modulation technique
PRA	Probabilistic Risk Analysis
PRE	Pre Processing
PSK	Phase Shift Keying
PROFIBUS	Process Field Bus
PWR	Pressurized Water Reactor
QD	Queuing Delay
QoS	Quality of Service
RD	Retransmission Delay
RPV	Reactor Pressure Vessel
RT	Real Time
RTEMS	Real Time Executive for Multi-Processor System
RTL	RT Linux
RTS/CTS	Request to Send/Clear to Send
RTOS	Real Time Operating System
SCWR	Super Critical Water Reactor
SD	Start Delimiter
SDA	Send Data with Acknowledge
SDN	Send Data with no Acknowledge

SIFS	Short Inter-Frame Space
SP	Smith Predictor
SPARC	Scalable Processor Architecture
SRD	Send and Request Data with Acknowledge
SS	Software Status
TCP/ IP	Transmission Control Protocol Internet Protocol
TD	Transmission Delay
TH_{max}	Threshold Maximum
TH_{min}	Threshold Minimum
RTR	Remote Transmission Request
WLAN	Wireless Local Area Network
WNCS	Wireless Networked Control Systems

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Reliability of the control system in a chemical and non nuclear plant depends on its communication medium and the protocols used to communicate between sensor actuator and controller node. A communication model was analyzed for this case study of a CANDU SCWR hydrogen co-generation unit that can efficiently produce hydrogen. In this case study, both nuclear and hydrogen cycles are coupled through a single channel steam flow and are housed in different buildings: therefore, an efficient real time networked communication system is essential for a plant's safe operation. Distances play a key role in communication delays. The networked system delay analysis and methods to minimize these delays can reduce communication errors. A wired network is a reliable communication medium that has been used in industry for decades. Wireless protocols, devices and standards have emerged as a competitive option for industrial instrumentation and control of plant's parameters. Wired networks require frequent maintenance due to harsh working environments in nuclear and chemical plants and need regular replacement to ensure safe and reliable operation; whereas, wireless networks using air medium and access points require less maintenance and can be used in specific areas of nuclear and chemical plants. Although wireless network adds redundancy to existing communication medium in plants, there are also present challenges such as hardware, software, and interference delays. Scarcity of frequency spectrum and the abundance of wireless devices has led to problems of signal interference and contention, which resulted in data corruption and delays. This research analyzed a network delay model as a communication model integrating hardware latency and software context

switching, system queuing and scheduling delays for the communication network of the co-generation model.

1.1 Background

Recent advancements in micro-chip technologies and software support have enhanced the system's performance, but there is also a growing need to evaluate the combinational reliability of the control system's software and hardware. In the past, various methods were used to evaluate the control system's performance, such as Failure Mode and Effect Analysis (FMEA) [1], Failure Mode Effect and Criticality Analysis (FMECA) [2], Preliminary Hazard Hnalysis (PHA) [3], Fault Hazard Analysis (FHA) [4] and Fault Tree analysis (FTA) [5]. Amongst various risk assessment methodologies the dynamic flowgraph methodology (DFM) [6-9] is an advanced approach for a dynamic system reliability analysis.

Recent work on DFM methodology [10-13] has suggested the use of the DFM model for the hydrogen production unit but to the best of this researcher's knowledge, a full scale DFM model for software and hardware network communication system as used in this study is modelled and simulated for the first time. A dynamic system can be modeled and analyzed for reliability and safety through the DFM. In a dynamic system, the DFM model, the variables and control system are represented through a time based logical cause and effect relationship. The results generated through DFM analysis are multi valued discrete events that can be initial, intermediate, and final events. These set of events can generate an unanticipated condition due to software logic errors, hardware failures, and environmental conditions or anticipated conditions representing the system behaviour.

DFM methodology develops a probability model based on the system known behaviour and requirements. A combination of software, hardware and communication channels are analyzed for sensor, actuator block communicating through a communication channel using standard Institute of Electrical and Electronics (IEEE) protocols and Open Systems Interconnection (OSI) model layers one, two and seven. Various combinations of inputs, associated conditions for both hardware and software, and timed step functions can be used in the DFM methodology. This research used software dymonda [14], which utilizes the DFM methodology deductive and inductive analysis. The deductive analysis generates prime implicants as output that can result in a top (failure) event; this case study is a combined delay of all the components of software, hardware, and communication channels. Deductive analysis generates the top event which is normally a failure state, using all intermediate nodes possible conditions. The conditions are set for both normal and unexpected values and results generated and colour coded to reflect the intensity of the unexpected conditions. Colour codes such as red, yellow, blue and green are used to model the intensity of the failure. The order is set with red being the most critical, yellow critical, green as permissible, and blue as moderate.

As mission critical system behaviour is difficult to analyze while it is operational, the DFM methodology generates results which present a complete picture of risks associated with every possible conditions of the hardware, software and communication protocols. The dymonda software has a built in probabilistic feature that ease the assessment of the system model in review. Using an iterative approach a full scale probabilistic risk assessment model that reflects of all the failure modes of the nodes for all possible combinations of inputs can be developed. The DFM model supports both forward and

reverse engineering approaches with the deductive analysis being as the reverse engineering approach while the inductive analysis being the forward engineering approach**.**

1.2 Motivation and Objectives of Research

The research objective is to analyze the application of a wireless communication in a CANDU SCWR hydrogen production case study through the DFM model.

Probabilistic Risk Assessment (PRA) techniques play a key role for design, operation and decision support system in chemical and nuclear power plants. Risks are normally associated with software and hardware failures and the DFM methodology represents a system's software and hardware failure states that lead to a top event (Failure event). For a failure condition, the intermediate states need to be identified, and preceding steps after a failure mode should also be determined to minimize risks. DFM methodology can represent all the system's possible conditions through decision table, which is a combination of all possible normal, and exceptional states associated with a time variant step function for preceding system's states.

The focus is on hardware (interrupt latency) and software (context switching) delays, transmission delays, and frame delays. For software computation, Flynn's taxonomy [15] for parallel computing networked communication model was used. The delay problem for real time control system was addressed by using the dymonda software that describes both the system hardware and software conditions. This study's contribution is in using a full scale CANDU SCWR hydrogen integrated co-generation model that covers all of the system communication components and a twofold historical and predictive analysis to

produce the system requirements, abnormal conditions and results that helps in model probabilistic risk assessment. Previous work focused on hard wired SCADA systems [16] that are increasingly redundant and though reliable yet lack the features of dependability in the case of a complete system failure and also require replacement or repair on frequent basis. Previous research [17] has shown that combining smaller packets into larger packets decreases the power consumption of wireless nodes. This research has shown that larger data size packets overhead decreases due to the comparatively larger size of the packets with respect to header and footer information. Larger packets can be efficiently transmitted using dual channel which also can improve link utilization and deceases delay. The maximum weight matching virtual output queue follows the principal of Largest Queue First (LQF) iterative method. The simulations demonstrated that LQF used in Virtual Output Queue (VOQ) bear lesser average delays than the Parallel Iterative Matching (PIM) method. The virtual software client server communication model uses thread switch control between a single server and multiple clients using VMBus in user, kernel and physical mode. Virtual application (Client) and VM Worker (Server) reside in user mode and communicate using virtual IP and subnet mask addresses within a cluster of client and server. A single program uses the parent-child control defined by global and local control in the matlab.

This research thesis is presented into seven sections. In section one a brief introduction of the delay analysis methods is discussed with description of contents in the following sections. Section two describes the copper chloride CANDU SCWR hydrogen co-generation model and various protocols for wireless communication. In section three delay analyses of wireless protocols are discussed. Section four discusses the comparative

delays of protocols and section five discusses their computations and results. In section six a comparative analysis of the CANDU SCWR hydrogen co-generation model, oxygen and hydrogen reactor model and the communication delay model is carried out through dynamic flowgraph methodology (DFM) and fault tree methodology. Chapter seven suggests future research trends.

2 DELAY ANALYSIS OF NCS

Communication medium, protocols, bandwidth and nature of communication traffic plays an important role in communication delays. This model assumes a communication between sender and receiver. For industrial networks the nodes are usually termed as sensor, actuator and controller nodes; therefore, the generic term node will be used in this communication model, so any of these three can be under consideration. This delay analysis focuses on the network communication Open Systems Interconnection model (*OSI model)* application layer, Media Access Control *(MAC)* layer, and physical layer delays. For this research, a non-pre emptive queuing delay and virtual output queuing delay analysis was performed.

2.1 Network and Communication Delays

Networked communication system delay analysis is important to evaluate the system reliability. The delay can be divided into three parts, which includes, the delay at the sender, the delay at the communication medium, and the delay at the receiver. The blocking delay is due to co-channel signal interferences from adjacent networks (Retransmission), plus the wait time for the buffered queue. The delay can be further divided as the difference between the time at which the sender sends the signal and the time at which the receiver receives the signal. NCS has sensor, actuator and controller nodes and if the time is denoted when the signal is transmitted by the sensor as T_S and signal received at controller as T_C then the elapsed time between the sensor and controller is the actual delay denoted by T_{DSC}. Similarly the signal time notation for two other nodes can be described as T_C and T_A and the corresponding delay between controller and actuator as T_{DCA}.

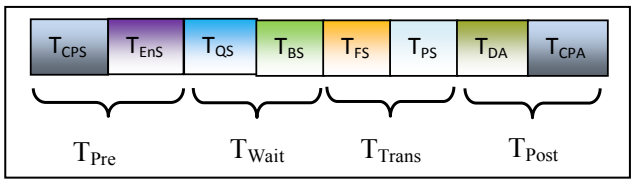


Fig. 2.1.1 Cumulative Delay from Sender to Receiver

$$T_{Pre} = T_{CPS} + T_{EnS} \quad (1)$$

$$T_{Wait} = T_{QS} + T_{BS} \quad (2)$$

$$T_{Trans} = T_{FS} + T_{DS} \quad (3)$$

$$T_{Post} = T_{CPC} + T_{DnC} \quad (4)$$

$$T_{DSC} = T_S + T_C \quad (5)$$

$$T_{DCA} = T_C + T_A \quad (6)$$

Here the time at both sender and receiver are defined. The receiver is identified as pre-processing (T_{Pre}) and the receiver is identified as post-processing (T_{Post}). Assuming that the sensor is sender node and the controller is the receiver node, thus the first equation is associated with sensor and the fourth equation is associated with the controller. Every node requires some time to compute (T_{CPS}, T_{CPC}), and then encode/decode (T_{ENS}, T_{DnC}) the data bits before transmission, as stated in equation one for sensor and equation four for controller. In equation two the wait time, queuing and back off timer involve wait time and it depends on the network traffic, contention and non contention based, and the queuing policy defined for a node or network. For this model non-prioritized, non-pre-

emptive single queue model and standard back off timer adapted for a collision scenario are used. In the equation three transmission delays are mentioned, which depend on the protocol frame size and data size associated with that particular protocol. Every protocol has data size limitation ranging from minimum to maximum size and pre-defined frame size. In a networked communication system (wireless sensor networks), the nodes normally operate in three states: idle, sleep and active. Whenever the sensor is not receiving any signal, it goes to an idle state and if it remains idle for a longer period of time, it goes to a sleep state. In this model, the sensor is the sender and in the case of idle, sleep and active states, there is a pre-processing time for the sender to a switch state and to transmit to receiver. The normal operation in a plant involves dealing with physical values such as temperature, pressure, and flow rate. A sensor node periodically sense data and forwards the data to the controller by converting it to a digital format. The controller evaluates the input signal from the sensor node and generates results based on a control algorithm and then forwards the results to the actuator to perform the task according to the control signal received. In Fig. 2.1.1, it is assumed that the sensor node is transmitting to the controller; therefore, T_{Pre} is the time taken by the sensor for pre processing and it is the sum of computation time T_{CPS}, and the encoding time T_{EnS}. Similarly the waiting time is T_{Wait}, which is the sum of queuing time at sensor buffer T_{QS} and the blocking time for any retransmission T_{BS}. The transmission time T_{Trans} depends on the message frame size T_{FS} and time taken by the packets to propagate that is T_{PS}. Finally the post processing time T_{Post} is at actuator/controller node and is the sum of computation time at controller that is T_{CPC} and decoding time at the controller that is T_{DC}. The computation and encoding/decoding delays depend on the operating system, and are independent of

protocol or communication medium; therefore they are fixed for a specific system. Queuing and propagation delays depend on the communication medium, while the frame and blocking delays are protocol specific. These delays can be expressed in terms of blocking time (for retransmission).

Finally the time delay between the sensor and controller is the time summation at which the sensor (T_S) sends the signal and the controller receives the signal (T_C). Similarly the time delay between the controller and actuator is the time sum at which the controller (T_C) sends the signal and the actuator receives the signal (T_A). PROFIBUS is deterministic communication protocol that uses token passing mechanism to communicate over a wired or wireless medium. Device Net is a dedicated carrier sense message priority communication protocol. For Device Net and PROFIBUS, the propagation delay is very small therefore this delay for the two networks is neglected, therefore the transmission delay T_{Trans}, now is only the frame delay that is identified as T_{FD}.

$T_{Trans} = T_{FS}$

Expected blocking delay $E\ [T_B]$ for Ethernet is expressed in [18] as:

$$E\ [T_B] \sum_{i=1}^{16} E\ [Ti] + T_{wait} \quad (7)$$

T_{wait} is the node wait time for the network to become free, where $E\ [Ti]$ depends on the number of participating nodes and rate of packet arrival at each node, and is the expected time for ith collision. Packets are accepted and then discarded after 15th collision [18].

Similarly for Control Net the blocking time is developed [18] as:

$$T_{min} = \sum i \square j \min(T_{Tx}, T_{Node}) \quad (8)$$

$$\sum i \square j \min(T_{Tx}, T_{Node}) = T_{Frame} + T_{Propagation} \quad (9)$$

$$T_{min} = T_{Frame} + T_{Propagation} \quad (10)$$

$$T_B = T_{wait} + \sum i \square j T_{Token} + T_{min} + T_{guard} \quad (11)$$

The time evaluated in the above equation shows the residual time T_{wait} already discussed above, T_{guard} is the guard time between two consecutive frames, while T_{Token} is the transmission time that a node can transmit withholding the token and finally T_{min} is the minimum time required for a frame transmission between nodes, including the frame data and the propagation time.

The device net blocking time is the sum of the residual time, and the total time taken to send high a priority node packet, $T_{Priority}$, over a period of jth message, with T_{bit} as the start bit and T_{Period} as the time period for the priority message.

$$T_{Priority} = \sum i \square j \left\lceil \frac{T_{B\,(k-1)+}\, T_{bit}}{T_{Period}} \right\rceil \quad (12)$$

$$T_{B(K)} = T_{wait} + T_{Priority} + T_{gaurd} \quad (13)$$

It is worth noting at this time that the frame size of protocol depends on data size in bytes, size of padding bytes, overhead bytes and if permissible the stuffing bytes. The values described here are protocol specific. Similarly, the signal propagation time depends on the distance between the communicating nodes and the bandwidth capacity of the transmission medium.

2.2 OSI Application Layer Delays

Communication and medium delays are dominant in networked control systems. In modern networked control, systems software also plays a key role in monitoring and controlling the equipment and parameter. So far we have discussed the delays involved in physical communication medium and protocols that operate at the physical and the MAC (Media Access Control) data link layer. The entire networked control system has hardware and software interfaces. These interfaces require Interrupt Requests (IRQ) or interrupt routines related to hardware which can cause processing delays. Hardware and software communication pass through these IRQs, that are pre-defined interrupt handlers to avoid any conflict between hardware units.

Several open source software programs are available online, that can be used for control in networked control systems. Real Time LINUX (RTL) and the Real-Time Executive for Multiprocessor Systems (RTEMS) features are open source Application Programming Interface (API) such as Portable Operating System Inerface (POSIX), supporting TCP/IP stack and Windows and Linux platforms. Both RTEMS and RTLINUX can operate on any standard CPU architecture such as Intel i960, MIPS, PowerPC, SPARC.

The RTEMS [19] is application layer based real time embedded software. The RTEMS is a cross platform real time operating system comprising coding and standard programming modules in its libraries. These libraries can be used for any standard application layer task. Language bindings of RTEM are in C and Ada with standard features of Real Time Operating Systems (RTOS) such as multi tasking, synchronization, schedulers etc. The RTEMS also supports multiple platform network computation on wired and wireless TCP/IP protocol stack. The standard hardware for RTOS include AMD, Intel, SPARC, Hitachi, Dell and RISC based systems.

2.3 RTLINUX

Real Time Operating System (RTOS) software operating at Kernel mode can override hardware IRQs (Interrupt Requests) [20]. The RTL follows a layered architecture separating the real time critical and non-critical modules, with real time critical tasks working in standard kernel shell. The separation involves periodic scheduling of tasks that can minimize the delay by prioritizing the real-time critical and non-critical tasks using threading techniques such as pthread. Pthread specifies a set of interfaces for threaded programming commonly known as POSIX threads. A single process can contain multiple threads, all of which are executing the same program. These threads share the same global memory, but each thread has its own stack. The approach to minimize the software delay is two layered, that is, first prioritize the functions of OSI application layer software over the operating system and then address the critical modules.

2.4 Software Response Time Delays

Physical, data link and application layer communication creates a response time delay. For real time sensor and control networks these delays can be important in network design and performance analysis. Response time is the actual system reaction corresponding to any system condition. As software in real time operating systems is used for controlling the hardware, it needs to access the interrupt handler for the particular hardware. The time lapse between an interrupt call and the system dispatching the interrupt is the interrupt latency. Scheduling the tasks on a priority basis is a built in function for a real time operating system. In priority processing there is a need for saving the current task context and restoration of new prioritized job. This delay due to the

priority sequence is called the context switch delay and its value depends on the system load and network controlled system processing power.

For both latency and context switching software delays used are the pre-calculated data of a 300 MHz PC [21] on Real time Linux (RTL) operating system.

$T_{RTL\ (Idle)Max} =$ 13.5 (Latency)

$T_{RTL\ (Idle)Max} =$ 33.1 (Context Switch)

$T_{RTL\ (Idle)Avg} =$ 1.7 (Latency)

$T_{RTL\ (Idle)Avg} =$ 8.7 (Context Switch)

$T_{RTL\ (Busy)Max} =$ 196.8 (Latency)

$T_{RTL\ (Busy)Max} =$ 193.9 (Context Switch)

$T_{RTL\ (Busy)Avg} =$ 5.4 (Latency)

$T_{RTL\ (Busy)Avg} =$ 15.4 (Context Switch)

The above delays are average and maximum values expressed in micro seconds.

Now the pre-processing and post processing delays are:

$$T_{Pre} = T_{CPS} + T_{EnS} + T_{Lat} + T_{Cont} \quad (14)$$

$$T_{Post} = T_{CPA} + T_{DeA} + T_{Lat} + T_{Cont} \quad (15)$$

Where T_{Lat} is latency delay T_{Cont} is context switching delay.

3. NCS PROTOCOLS

Wireless communication standards and protocols are continuously reviewed by various IEEE working groups and several standard protocols are used in wireless networked control systems. Most of these protocols work within Open Systems for Interconnection (OSI) data link layer and they can be classified on the basis of the system access mechanism, as a *deterministic allocation mechanism* such as Time Division Multiple Access (TDMA) that is Point Coordination Function (PCF), Code Division Multiple Access (CDMA) and Frequency Division Multiple Access (FDMA); random *allocation mechanism* which is ALOHA and Ethernet, and *demand allocation mechanism* that is a Token passing.

ALOHA was the first pure computer network protocol and its mechanism for transmission is simple. A sender can transmit data without worrying about the contention and in case of collision; the sender must retransmit the same data. ALOHA performance can be measured by the number of frames transmitted successfully (throughput) over a period of time. ALOHA was established in the early times of computer networks with few computers and limited network traffic. The growth in computer networks and internet invention ALOHA is replaced by Ethernet, Token passing and Zigbee protocols.

3.1 Ethernet (CSMA/CA-CD)

Ethernet Carrier Sense Multiple Access-Collision Detection (CSMA/CD) is a well established standard for both wired and wireless communications. IEEE Ethernet uses 802.3 and 802.11 standards for wired and wireless network communication. Carrier sensing is simple in IEEE 802.3 wired networks. The sender senses the peak voltage on

the wired network and compares it to a predetermined threshold voltage before communicating. If the sensed voltage is greater than the threshold then a communication collision has occurred. In order to communicate, the sender would continue sensing the medium until the sensed voltage is lesser than the threshold voltage. In IEEE 802.11 the OSI physical layer sensing is accomplished by a Clear Channel Assessment (CCA) signal that is integrated in the Physical Layer Convergence Protocol (PLCP). The carrier sensing can be done by either detecting the bits on the communication channel or by evaluating the Received Signal Strength (RSS) against a predetermined threshold value. Values above the threshold indicate channel interference.

An Ethernet IEEE 802.11 wireless communication access mechanism can either communicate by a Distributed Co-ordination Function (DCF) that is a Request to Send (RTS) Clear to Send (CTS) mechanism, or by a Point Coordination Function (PCF).

The basic 802.11 MAC protocol is the Distributed Coordination Function (DCF) that operates on the principle of listen before sending. This listening time is the standard back off time and is called the DCF Inter frame space (DIFS). In case the communication medium is idle and the sender node wants to send packets to the receiver node, it first sends an RTS (Request to Send) packet to the receiver node. After receiving the RTS packet, the receiving node senses the network condition for a Short Inter-frame Space (SIFS) time and then acknowledges by clear to send a clear to send (CTS) packet. Upon receiving the CTS packet, the sender transmits data to the receiver after waiting for SIFS space and on successful transmission the sender acknowledges ACK after SIFS space. In the event of a collision, the CTS or ACK packets are not received and the sender has to retransmit.

Each participant idle node in the network maintains a Network Allocation Vector (NAV) for deferring its transmission during the RTS, the CTS and the Data communication phase.

Figure 3.1.1 shows the transmission of data packets in a sequence described as below.

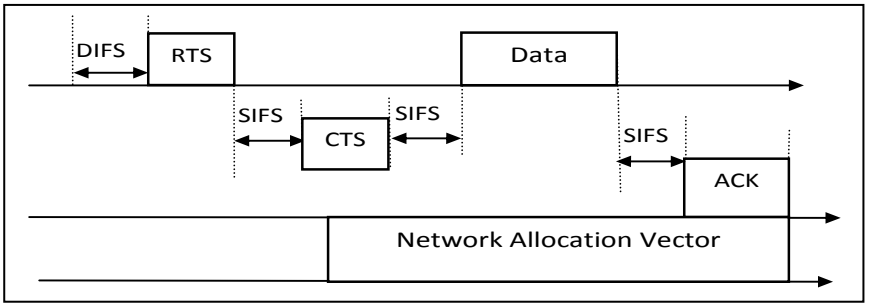


Fig. 3.1.1 Ethernet DCF Operation

If two nodes transmit simultaneously, a collision will occur that would result in the packets corruption. The nodes have to wait to retransmit for a period equal to the contention window, C_W. The contention window size depends on the number of the frame transmit failure. For every new collision the size of the contention window increases proportionally to the Binary Exponential Back off (BEB) mechanism. The contention window size increase is shown in Fig. 3.1.2.

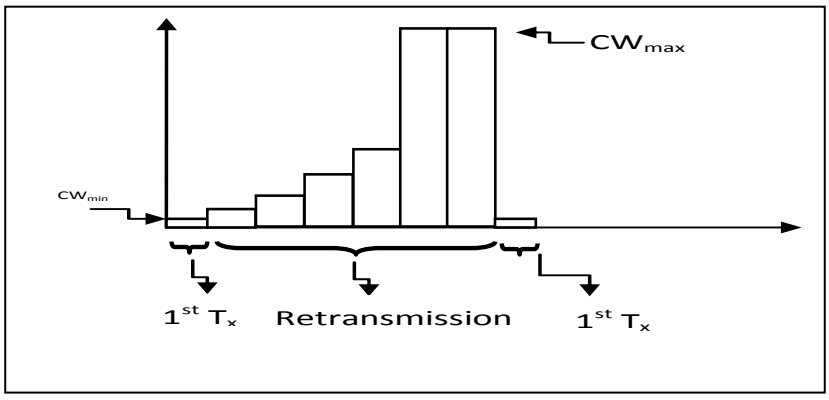


Fig. 3.1.2 Contention Window Exponential Back off

Binary Exponential Back off selects a random slot for transmission which is from 0 to CW_{min}. Where CW_{min} is the minimum contention window size in the order of 2^n -1, and the preceding windows are in the order of 2^{n+1} -1 and so on. The last retransmission window is the largest contention window CW_{max} , equal to 1023.

In the two way hand shaking mechanism (without RTS and CTS), the delay is noted as

$$T_{Delay} = DIFS + Data + T_{Prop} + T_{SIFS} + ACK + T_{Prop} \quad (16)$$

and for the four way hand shaking mechanism with RTS and CTS, the delay is

$$T_{Delay} = DIFS + RTS + T_{SIFS} + T_{Prop} + CTS + T_{SIFS} + T_{Prop} + Mac + Phy + Data + T_{SIFS} + ACK + T_{Prop} \quad (17)$$

In case of a collision the delay can be summed as

$$T_{Delay} = DIFS + RTS + T_{SIFS} + CTS \quad (18)$$

For these calculations the assumption is no collision.

$$\text{Throughput} = \frac{Data\ Size\ in\ Bits}{DCF\ or\ PCF\ Delay + Protocol\ Overhead} \quad (19)$$

$$\text{Efficiency } (\square) = \frac{Throughput}{Data\ Size} \quad (20)$$

Table 3.1.1 is a comparative analysis of access time delays for various 802.11 standards. Frequency Hopping Spread Spectrum (FHSS) involves a frequency hop transmit sequence for data transmission within a narrow band carrier frequency. The carrier frequency is periodically modified following a specific sequence of frequencies, often termed as spreading code. A wide band signal is generated if the FHSS is spread over a period of seconds.

The Direct Sequence Spread Spectrum (DSSS) uses the Phase Shift Keying (PSK) modulation technique to convert the data into bit symbols, known as chip sequence. The chip sequence is in the form of 0 and 1 bit. Both FHSS and DSSS are 802.11, radio frequency LAN standards operating at the OSI, physical layer.

Infra red communication was not considered for the delay analysis due to its short operating range. The infra red protocol for a comparative analysis was mentioned. A review of 802.11 various versions, such as a, b, e, g. was conducted. The 802.11g/e has higher data rates and higher throughput. It functions on Orthogonal Frequency Division Multiplexing (OFDM) which is a wideband communication scheme. The OFDM guard band minimizes the Inter Symbol Interference (ISI).

IEEE 802.11 Parameters						
Description	**802.11 FHSS**	**802.11 DSSS**	**802.11 IR**	**802.11 b**	**802.11 a**	**802.11 g/e**
t_{slot}	50µsec	20 µsec	8 µsec	20 µsec	9 µsec	9 µsec
SIFS	28µsec	10 µsec	10 µsec	10 µsec	16 µsec	10 µsec
PIFS	SIFS + t_{slot}					
DIFS	SIFS +(2 x t_{slot})					
CW_{min}	15	31	63	31	15	15
CW_{max}	1023	1023	1023	1023	1023	1023
$D_{maxRate}$	2 Mbps	2 Mbps	2 Mbps	11 Mbps	54 Mbps	54 Mbps
$f_{operation}$	2.4 Ghz	2.4 Ghz	850-950 nm	2.4 Ghz	5 Ghz	2.4 Ghz
Throughput Max.	1.2 Mbps	1.2 Mbps	N/A	5.5 Mbps	32 Mbps	24 Mbps
Physical Layer	DSSS	FHSS	N/A	OFDM	OFDM	OFDM

Table 3.1.1 IEEE Ethernet Standard Parameters

Fig. 3.1.3 shows the IEEE Exponential Back off execution, and the 802.11 standard defines that exponential back off algorithm executes if and when the node senses a busy medium, after each retransmission and after every successful transmission. The only scenario when the back off algorithm is never used is the system being idle for more than DIFS time.

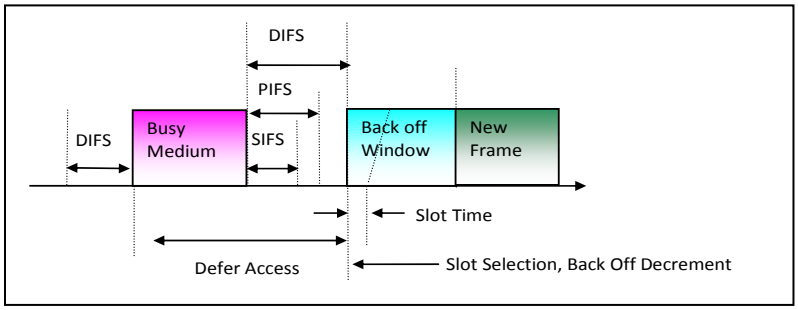


Fig. 3.1.3 Exponential Back off Execution

The 802.11 Infra Red (IR) is an optical signal propagation method that uses the Pulse Position Modulation (PPM) technique, operating at the 800-900 nm band, which is well suited to counter the signal propagation losses during interference.

The 802.11 a /b are 802.11 data link layer protocols, with enhanced data rates. The 802.11a can carry data up to 54 Mbits/sec operating at 5GHz band. The 802.11b can carry data up to 11 Mbits/sec operating at 2.4 GHz band. The 802.11a has a narrower coverage range than 802.11b, due to the higher carrier frequency and smaller wavelengths; on the other hand 802.11b operates on widely used 2.4 GHz band and can experience interference from adjacent devices such as microwave or hand held devices.

For 802.11 wireless communications the data transmission time is divided into Contention (CP) and Contention Free Period (CFP). Real time system requires

deterministic and error free communication; the Point Co-ordination Function (PCF) is considered suitable for a real time communication system as it operates in contention free time period. The Fig. 3.1.4 explains the operation of DCF and PCF during Contention period (CP) and the Contention Free Period (CFP). B is the PCF beacon frame and NAV is the Network allocation Vector.

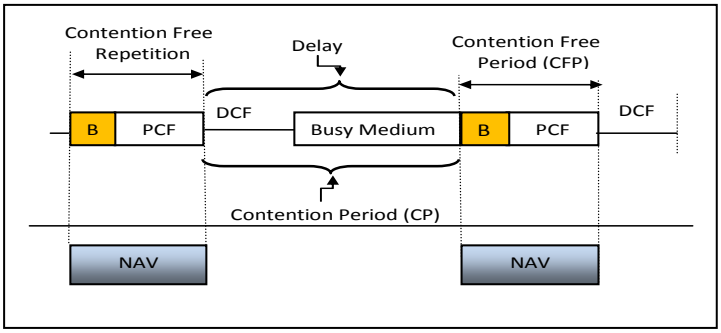


Fig.3.1.4 PCF Contention Free window

In IEEE 802.11 wireless carrier sense medium access protocols, exposed and hidden terminal problems are well known. They can cause packet collision and bandwidth under utilization which contributes to network transmission delays.

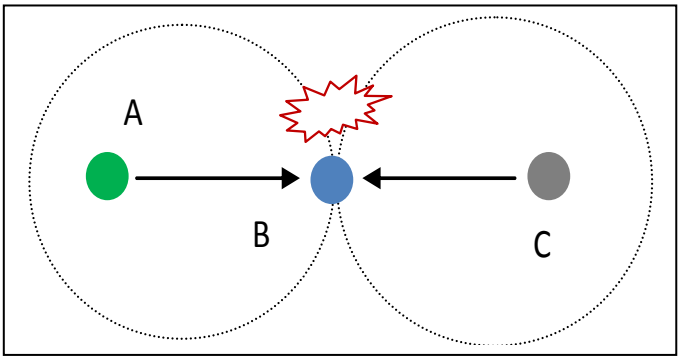


Fig. 3.1.5 Hidden Node Problem Causing Collision

Fig. 3.1.5 shows the transmission ranges of nodes A, B, and C. Node B is in the transmission range of both node A and C; similarly A can transmit to B but not to C, and C can transmit to B but not to A. Problems arises when both A and C transmit to B node simultaneously as A and C are out of range and hidden from each other. The simultaneous transmission would result in collision. The hidden node problem can be resolved by using centralized communication topology with one intelligent master node controlling the slave nodes for coverage area to control its collision. Routing path and vicinity information can be shared by the master nodes to avoid collision. Similarly, the same strategy can be adapted for exposed nodes problems.

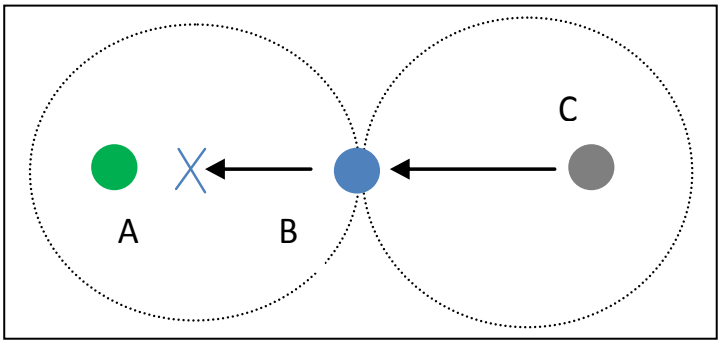


Fig. 3.1.6 Exposed Nodes Problem Causing Bandwidth under Utilization

In Fig. 3.1.6, node C is transmitting to node B, although node B can transmit to node A, but after sensing node C, it ceases it`s transmission to node A. Although there would be no contention, however B can only transmit when it is not receiving any other transmission. The result is bandwidth under utilization and unwanted transmission delay.

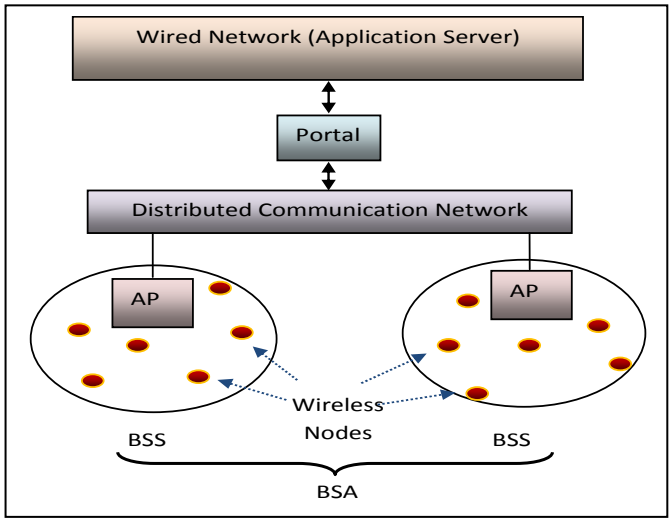


Fig. 3.1.7 Ethernet Communication Network

Fig. 3.1.7 shows a typical Ethernet infrastructure. The Ethernet networks have a dedicated server with specific computing and storing application. The server is connected to a gateway access portal, which is linked to the distributed communication network. The wireless nodes are serviced by their respective Access Points (AP). These APs are networked through a wired or wireless communication medium that is referred to as a Distributed System (DS). The nodes accessing the network can act as sensors, actuators, or a single node which can perform both functions. Each AP controls a set of Basic Service Set (BSS) comprising wireless nodes in its coverage area. All the BSSs within a network form the Basic Service Area (BSA).

3.2 Control Net (Token Passing)

A Control Net is a highly deterministic protocol standard used in industry for a real time Networked Control System (NCS). PROFIBUS (FIELD BUS), and Manufacturing Area Protocol (MAP) are common examples of control net. The PROFIBUS standard was established to replace the analog systems with the digital control and data communication

systems. PROFIBUS uses token passing network topology for transmitting control and data signals. The token network is formed by a set of nodes arranged in a logical polling tree hierarchy. The PROFIBUS architecture is a typical master slave architecture where slaves are polled by the master for token passing. The PROFIBUS has two data priority levels, high and low priorities which are controlled by master nodes.

This protocol ensures contention free data transmission and it also ensures a fair share of medium for every node. Every node has the knowledge of its predecessor and successor and can transmit over a specific period of time, and has to regenerate the token for the next node. The token is generated automatically if the node possessing the token cannot release it. Token passing is deterministic and contention free, but it is also a system overhead for large network traffic. There are four kinds of Field bus Data Link (FDL) services used in PROFIBUS:

a) SDA: Send Data with Acknowledge
b) SRD: Send & Request Data with Acknowledge
c) SDN: Send Data with No Acknowledge
d) CSRD: Cyclic SRD

In SDA, an Acknowledge (ACK) signal is required between master and slave nodes.

In SRD, a Data ACK signal is required between master and slave nodes.

In SDN, no ACK signal is required between master and slave nodes.

In SRD, recycle data transmission with ACK signal is required between master and slave nodes.

Different message frame structures are used by PROIBUS for each FDL data transmission.

Token frames, action frames, and acknowledge frames are the standard message frames used by PROFIBUS.

Once a user has a token frame, it can send data over the bus using an action frame which contains fixed or variable length user data. Action frames contains stop delimiting characters, frame control area and frame checksum bytes besides starting delimiting character bytes of source and destination addresses. Acknowledge frame is the confirmation returned by the receiver within a regulated time length after receiving data from the token holder. This immediate response in PROFIBUS ensures real time message transmission which makes it suitable for real time safety critical communication. The ACK frame contains a starting character, source, and destination addresses, frame control area, frame checksum result, and stop delimiting character. A synchronization frame of 33-bits must be added before every action frame.

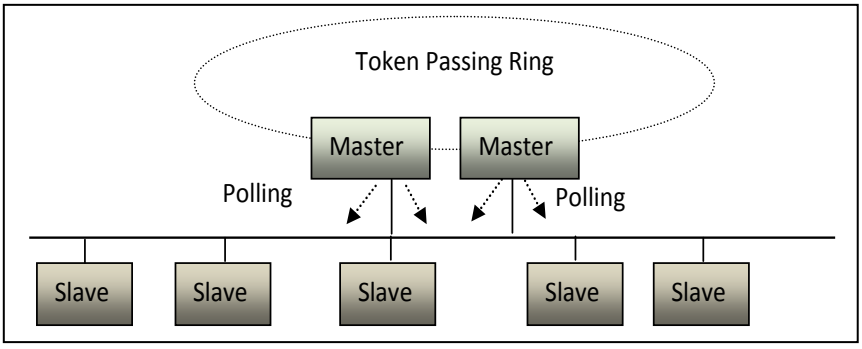


Fig 3.2.1 PROFIBUS (Tree) Communication Network

The PROFIBUS logical tree architecture is shown in Fig. 3.2.1. Master nodes form the logical tree hierarchy by polling the slave nodes for the token. The first node acknowledges that poll gets the token and transmits.

There are four types of frame formats in PROFIBUS: SD1, SD2, SD3 and SD4.

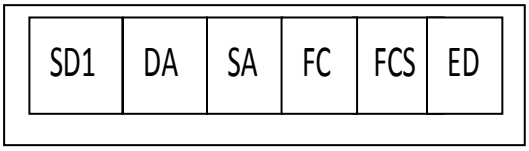


Fig.3.2.2a PROFIBUS SD1Frame

Fig. 3.2.2b PROFIBUS SD2 Frame

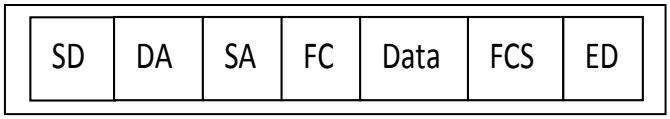


Fig. 3.2.2c PROFIBUS SD3 Frame

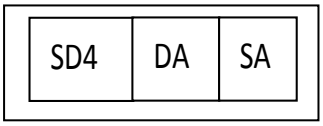


Fig. 3.2.2d PROFIBUS SD4 Frame

The acronyms used in the frames are defined as:

SD1 to SD4 are starting delimiting characters of 1 byte each. LE and LER determines the data length contained within the frame. DA and SA are destination and source addresses, while FC and FCS are frame control and frame checksum respectively. Data can be fixed or variable while ED is the end of the delimiter. SD1 and SD4 are only data free token

frames while SD2 and SD3 are fixed and variable length data frames respectively. The frame lengths from SD1 to SD4 can be determined by the number of divisions with each division of 11 bits each. Maximum allowable data size is 246 bytes, while the SD3 fixed data length is 88 bits. SD1 has six divisions and is 66 bits in length while the SD2 frame size depends on the data size. SD3 frame is comprised of 154 bits, with 88 bits for fixed data while SD4 is 33 bits in length. The PROFIBUS requires an additional synchronization time, which is actually the token frame used by the slave nodes to communicate. The nodes participating in the network need synchronization time. For calculations, data transmission size of N_B bytes combined with 9 stuffing bits were used. The slave node maximum response time to transmit data is $T_{Max\text{-}SLR}$. For data transmission computations, all four kinds of Field bus Data Link (FDL) services were used.

3.3 Device Net (CAN)

The Device Net is a deterministic protocol that is used in industry for real time data transmission. Device Net is commonly known as a Controller Area Network (CAN), which is a carrier sense access mechanism with message priority. Each message has a priority set by the 11 bit arbitrary identifier, with logic 0 having a priority over logic 1. The transmitting node waits for the idle bus to transmit its bit stream message. In case of simultaneous transmissions, both nodes float their message frames over the bus and listen. The node that receives the same bit as transmitted gets the right to transmit. The CAN is a multicast communication protocol, where all nodes receive the message frame from the sender node and accept or reject them on the basis of the filtering mask.

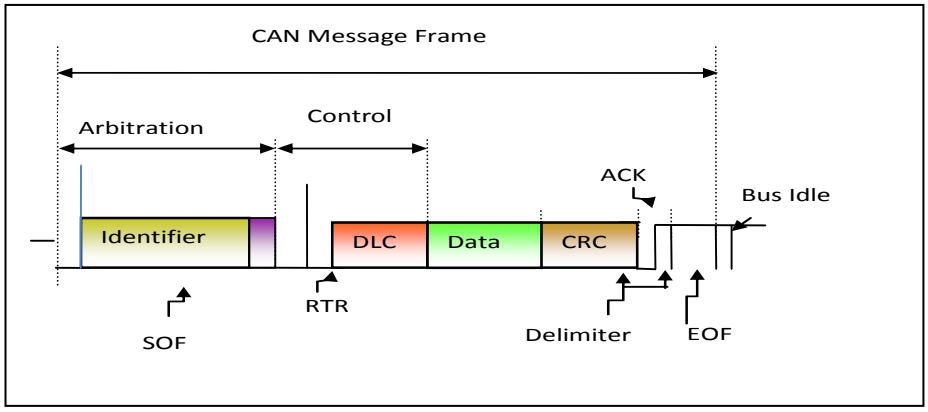


Fig. 3.3.1 Device Net Message Frame Format

The CAN message format is shown in Fig. 3.3.1. The message frame starts with a start of frame (SOF) bit, followed by an 11 bit identifier. Whereas the Data Link Control (DLC) field is followed by r_0 and r_1 initializing bits. The user data field is from 0 to 8 bytes in length. The Cyclic Redundancy Check (CRC) field is 15 bits in length. Acknowledgement (ACK) and End of Frame (EOF) marks the end of the message frame.

3.4 IEEE 802.15.4

IEEE 802.15.4 wireless communication is low data rate and low power consumption protocol. IEEE 802.15.4 is frequently used in NCS with star or peer to peer topology and operates in two Industrial Scientific and Medical (ISM) frequency bands, 915 MHz and 2.4 GHz.

As shown in Table 2, IEEE 802.15.4 is a standard for communication between two devices such as sensor and controller or between controller and actuator and works on the MAC (Medium Access control) layer of the OSI model. IEEE 802.15.4 operates on standard Direct Sequence Spread Spectrum (DSSS) technique that can be implemented using phase shift keying modulation. Networked control IEEE 802.15.4 sensor nodes

operate in three modes i.e. active, sleep, and idle. Although sleep and idle can optimize the power consumption problem, they induce an unwanted delay between the states which can be critical for real time system. Mobile Adhoc Networks (MANET) [22] dynamic topology can be adapted to counter these delays. Master-slave clusters with flexible knowledge based mode change between the nodes can tackle these delays. Zigbee operates on AODV (Ad hoc On-Demand Distance Vector) [23] routing protocol that can adapt node movement between adjacent nodes operating within the cluster of one master, that is a centralized node and various distributed slave nodes.

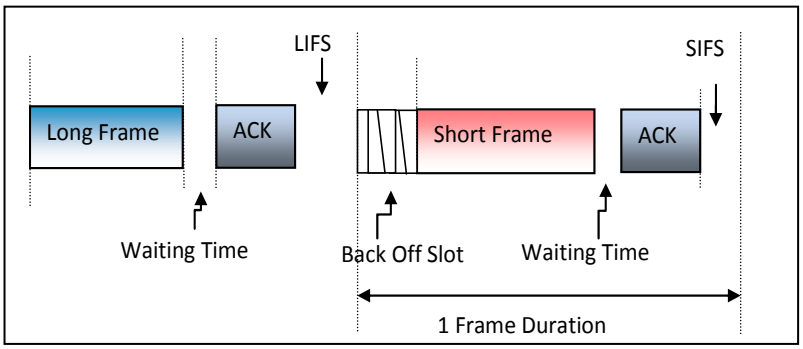


Fig. 3.4.1 Zigbee Frame Format

Fig. 3.4.1 shows the transmission format of Zigbee short/long frame sandwiched between back off time slot and waiting time. The transmission can be acknowledge or non-acknowledge based. The node association and disassociation are governed by the master node that updates its route configuration based on the signal's broadcasts and the ACK received. Any node that is not part of the cluster cannot communicate with the cluster nodes.

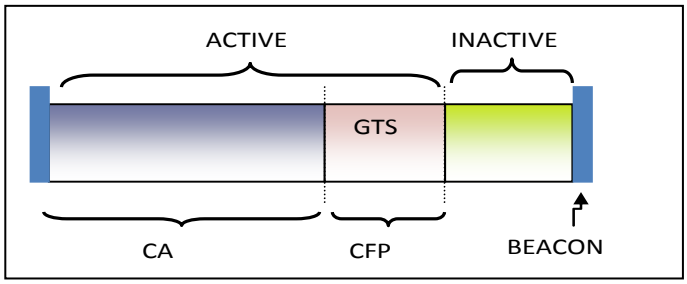


Fig. 3.4.2 Zigbee Beacon Enabled Mode Super Frame

Beacons intervals can be from 15 ms to 251 s and between these beacons the super frame has active and inactive periods. The inactive period is the idle mode in which no device can communicate.

Table 3.4.1 shows 27 communication channels for the physical layer, and they are, defined in the industrial scientific medical (ISM) band, which has 16 channels at 2.4 GHz band, 10 channels at 915 MHz, and 1 channel at 868 MHz. All of these channels operate at the Direct Sequence Spread Spectrum propagation mechanism. IEEE 802.15.4 is a low range and low power protocol and due to its neighbouring interference devices are expected to cover a 20- 30 m range, while without interference their range can be extended up to 100 meters. The range can be increased with the use of repeater or amplifier for a range up to 250-300 metres.

IEEE 802.15.4 Frequency Bands and Specifications		
Description	**Prescribed Values**	
	915 MHz	**2.4 GHz**
Raw data bit rate	40 kbps	250 kbps
Transmitter output power	1 mW	
Transmission range	With Interference: up to 30 m; Without Interference: up to 100 m	
Latency	15 ms	
Channels	10 channels	16 channels
Channel numbering	1 to 10	11 to 26
Channel access	CSMA/CA-CD	

Table 3.4.1 IEEE 802.15.4 Specifications

The delay equation can be written as

$$T_{Delay} = T_{slot} + T_{Frame} + T_{wait} + T_{LIFS} \text{ or } T_{SIFS} + ACK + T_{Prop} \quad (21)$$

$$T_{Frame} = \frac{\text{Phy} + \text{Mac} + \text{Data}\ (Short\ or\ Large\ Frame) + \text{Mac Hdr} + \text{Mac Ftr}}{Data\ Rate} \quad (22)$$

$$\text{Ack} = \frac{\text{Phy} + \text{Mac Hdr} + \text{Mac Ftr}}{Data\ Rate} \quad (23)$$

Where data is dependent on frame size.

3.5 COMPARATIVE DELAYS

3.5.1 Ethernet

Table 3.5.1 shows IEEE 802.11 protocol delay. For further calculations, the study used equation 15 and 16 of Chapter 3 to compute delay values for the various classes of IEEE 802.11 delays.

IEEE 802.11 Delay Data	
Description	
Slot Time	20 μs
Propagation Delay	2 μs
RTS	352 bits
CTS	304 bits
ACK	304 bits
Channel Bit Rate	1 Mbps
CW_{min}, CW_{max}	32, 1024
PHY Header	24 bytes
MAC Header	28 bytes

Table 3.5.1 IEEE 802.11 Delays

3.5.2 PROFIBUS

PROFIBUS Delay Data			
Description			
SD1	6 slots		11 bits each
SD2	9 slots + 1 data slot		11 bits + 1 variable (Data Max = 246 bytes)
SD3	6 slots + 1 data slot		11 bits + 88 bits
SD4	3 slots		11 bits each
Token	Tsync	Slots	11 bits each
	3	3	

Table 3.5.2 PROFIBUS Delays

The token frame size (SD4) without any data can be calculated as:

$(TSYN + 3) \times 11 = (\ 3+3\ ) \times 11 = 6\times 11=66$

For data transmission the SDA frame size was calculated.

$SDA\_T_{frame} = T_{SYN} + (\ Nb+9\ ) \times 11 + T_{max\text{-}SLR} + 6\times 11$

Similarly for the SDN frame

$SDN\_T_{frame} = T_{SYN} + (\ Nb+9\ ) \times 11$

SRD_T_{frame} = T_{SYN} + 6× 11+ $T_{max\text{-}SLR}$ + (Nb+9) × 11

Assuming slave node response time is 200bit and data transmission speed is 1.5Mbit/s and Nb is 8 bytes.

T_{frame} =66/1.5=0.044μs

SDA_T_{frame} = (198+8× 11+200) /1.5=0.324μs

SDN_T_{frame} = (132+8× 11) /1.5=0.147μs

SRD_T_{frame} = (198+8× 11+200) /1.5=0.324μs

If Nb is 246Bytes, then

T_{frame} =66/1.5=0.044 μs

SDA_T_{frame} = (198+246× 11+200) /1.5=2.069 μs

SDN_T_{frame} = (132+246× 11) /1.5=1.892μs

SRD_T_{frame} = (198+246× 11+200) /1.5=2.096 as

3.5.3 Device Net

Table 3.5.3 illustrates the device net delays and assumes 1bit equal to 1 micro seconds.

Device Net Delay Data	
Description	
Start of Frame (SOF)	1 bit
Identifier + RTR	11 bits
RTR	1 bit
Control bits	6 bits
Data	0 to 8 bytes
CRC	15 bits
CRC Delimiter	1 bit
ACK	2 bits
ACK Delimiter	1 bit
EOF	7 bits

Table 3.5.3 DeviceNet Delays

3.5.4 Zigbee

Equation 21 was used to produce data shown in Table 3.5.4.

IEEE 802.15.4 Delay Data	
Description	
Slot Time	20 μs
T_{wait}	2 μs
T_{LIFS}	192 μs
T_{SIFS}	640 μs
ACK	304 bits
T_{Prop}	1 Mbps
$Data_{Short\ Frame}$	≤ 18 bytes
$Data_{Large\ Frame}$	>18 bytes

Table 3.5.4 IEEE 802.15.4 Delays

3.6 Queuing Delays

A queue service model is the delay and priority mechanism that determines the packet's arrival rate, packet service rate, and packet's priority.

λ = Average packet arrival rate

μ = Average packet service rate

c = Transmission link capacity

s = Packet size

ρ = Link utilization factor

Such that

$$\mu = \frac{c}{s}$$

$$\rho = \frac{\lambda}{\mu}$$

The packet arrival process in queuing theory is modeled as a Poisson process. Each and every arrival is independent of every other arrival. They are memory-less; which makes all arrivals a special case of markov chain process. Poisson process has an exponential inter-arrival time (Delay) given as

$$E(t) = \int_0^t f_t \; d_t = \frac{1}{\lambda}$$

It is assumed that the packets are also exponentially distributed. For n number of random packets in the system, P_n is the probability of n packets in the system. In finite buffer state, n is equal to k, where k is the buffer size. For the buffer occupancy variation between n-1, n and n+1 states w.r.t time, the probability equation is

$$P_n(t+\delta) = P_n(t)\,[(1-\lambda\delta)\,(1-\mu\delta)] + P_{n-1}(t)\,[(\lambda\delta)(\,1-\mu\delta)] + P_{n+1}(t)\,[(\mu\delta)\,(1-\lambda\delta)] + 0(\delta)^{*} \quad (24)$$

Where δ is the service time distribution.

Given that

$$\frac{dPn\,(t)}{d(t)} = \frac{Pn\,(t)\,[(1-\lambda\delta)\,(1-\mu\delta)]}{\delta} \quad (25)$$

$$\frac{dPn\,(t)}{d(t)} = -\,(\lambda+\mu)\,P_n\,(t) + P_{n-1}(t)\,\lambda + P_{n+1}(t)\,\mu \quad (26)$$

In steady state

$$\frac{dPn\,(t)}{d(t)} = 0$$

Therefore

$$(\lambda+\mu)\,P_n\,(t) = P_{n-1}(t)\,\lambda + P_{n+1}(t)\,\mu$$

The above equation can alternatively be derived by equating system flux variations.

Consider a finite queuing model with maximum queue capacity k.

Mathematically

$$P_n = \rho^n\,P_o$$

Where

$$0 \leq n \leq k$$

The initial condition for i channels

$$P_o \sum_{i=0}^{k} \rho^i = 1$$

$$P_o = \frac{\rho_o\,(1-\rho)}{1-(\rho^{k+1})}$$

For n packets

$P_n = \frac{\rho_n (1-\rho)}{1-(\rho^{k+1})}$

For n packets equal to k buffer size

$P_k = \frac{\rho_k (1-\rho)}{1-(\rho^{k+1})}$

The above equation is also the buffer blocking state probability for a finite buffer

$P_{Blocking} = \frac{\rho_k (1-\rho)}{1-(\rho^{k+1})}$

The net arrival rate for a finite buffer is its thoroughput ζ, and is given as

$\zeta = \lambda (1 - P_{Blocking})$

The above equation is also true for system conservation. Considering the system output for the finite buffer markov chain we have

$\lambda (1 - P_{Blocking}) = \mu (1 - P_o)$

The equation for an infinite buffer is reduced as

$\lambda = \mu (1 - P_o)$

Now for buffer with $\rho < 1$, $\rho^{k+1} << 1$ (negligible), therefore

$P_n = \rho_n (1 - \rho)$

Similarly for buffer with $\rho >> 1$, $\rho^{k+1} >> 1$ (infinite), therefore

$P_n = \frac{\rho^{n+1}}{\rho^{k+1}}$ or $P_n = \rho^{n-k}$

It was assumed that the packets are exponentially distributed (E_n), and by Pollaczek-Khinchine formula, to have

$$E_n = \left(\frac{\rho}{1-\rho}\right)\left(\frac{\rho}{1-2}\right) + (1-\mu^2\delta^2)$$

$$E_n = \left(\frac{\frac{1}{\mu}}{1-\rho}\right)\left(\frac{\rho}{1-2}\right) + (1-\mu^2\delta^2)$$

Where δ^2 is the variance of service time distribution and $\frac{1}{\mu}$ is the mean service time. Any increase in δ^2 would increase the value of E_n. The traffic model is a poison arrival queuing model, that can be pre-emptive or non pre-emptive with the assumption of two non-prioritized classes of traffic for control and data transmission. Both of the class 1 and class 2, traffic are mixed together on one transmitter and through one out going link. The data is assumed as Poisson exponential distribution while control signal is assumed as poison fixed distribution.

λ = Total average packet arrival rate

λ_1 = Average data packet arrival rate

λ_2 = Average control packet arrival rate

μ = Total Average packet service rate

μ_1 = Average data packet service rate

μ_2 = Average control packet service rate

A similar model can be assumed for multiple channels or multiple paths. For two channels the link utilization factor comes to

$$\rho = \frac{\lambda}{2\mu}$$

The probability function for ‘o’ to ‘n’ packets

$$P_1 = 2\,\rho\,P_o$$

$$P_n = 2\,\rho^n\,P_o$$

$$P_o = \left(\frac{1-\rho}{1+\rho}\right)$$

$$P_n = 2\rho^n\left(\frac{1-\rho}{1+\rho}\right)$$

$$E_n = \sum_{i=0}^{k} \rho^i$$

$$E_n = \left(\frac{2\rho}{1+\rho^2}\right)$$

Take three cases for queuing delay analysis i.e multiple links with single service, single link with single service, and single link with multiple services. The delay ($E_{(t)}$) equation for the three cases can be written as

$$E_{(t)} = \frac{1}{\mu(1-\rho^2)}$$

$$E_{(t)} = \left(\frac{\frac{1}{2\mu}}{1-\frac{\lambda}{2\mu}}\right) = \frac{1}{2}\left(\frac{\mu}{1-\rho}\right) \qquad (27)$$

$$E_{(t)} = \left(\frac{\frac{1}{2\mu}}{1-\frac{\lambda}{\mu}}\right) = \left(\frac{\frac{1}{2\mu}}{1-2\frac{\lambda}{2\mu}}\right) = \frac{1}{\mu(1-2\rho)} \qquad (28)$$

The delay equation for case 2 (Eqn. 27); one link with double service rate can provide twice the transmission rate: thereby, minimizing the delay by half.

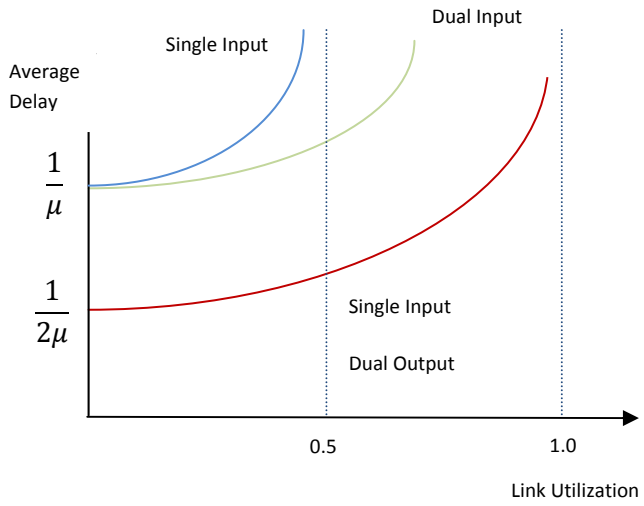


Fig 3.6.1 Queuing Delay for Two-Class

Pollaczek-Khinchine's formula for two class queuing model can be expressed as

$$E_n = \frac{\lambda(E\tau^2)}{2(1-\rho)} \quad (29)$$

Where $E\tau^2$ is the second moment of service time and is expected weighted sum of all $E_i^{\tau^2}$ such that

$$E\tau^2 = \rho^2 + \frac{1}{\mu^2} \quad (30)$$

Assuming data signals of 960 bits and control signals of 48 bits. Further assume the mean service time $\frac{1}{\mu_2}$ for data bits equal to 100 ms (milliseconds) and the mean service time $\frac{1}{\mu_1}$ for control bits equal to 5 ms (milliseconds). Average packet arrival rate for data signals is assumed as 80 per cent, which is 0.8λ, and the average packet arrival rate for control signals as 20 per cent, which is 0.2λ. Now total link utilization factor ρ is the

sum of the two utilizations for data and control signals, ρ_2 and ρ_1. For one way transmission, assume ρ as 0.5. From our assumptions to derive mathematical values as

$\rho = \rho_1 + \rho_2$

0.5 = 0.2λ (0.005) + 0.8λ (0.1)

λ = 6.17 packets / sec

As

$E\tau^2 = E_1^{\tau^2} + E_2^{\tau^2}$ (31)

$E\tau^2 = 0.2(\rho_1^2 + \frac{1}{\mu_1^2}) + 0.8(\rho_2^2 + \frac{1}{\mu_2^2})$

Where

$\rho_2^2 = \frac{2}{\mu_2^2}$ and $\rho_1^2 = 0$ (negligibily small)

Therefore

$E\tau^2 = 0.2(\frac{1}{\mu_1^2}) + 0.8(\frac{2}{\mu_2^2} + \frac{1}{\mu_2^2})$

$E\tau^2 = 0.024$

$E_n = \frac{6.17(0.024)}{2(1-0.5)}$

E_n = 148 milliseconds

The above delay is for non-prioritized pre-emptive and non-pre-emptive queues.

3.7 Scheduling Delays

Field programmable gateway array (FPGA) controllers are scheduling fabric used in controller networks that receives signals through a multi-channel input buffer and directs it to a multi-channel output buffer by using a novel algorithm. The scheduling depends on the type of traffic arriving at the controller. The scheduling can be done at input or output buffer. The assumption that all FPGA arrivals are Bernoulli [24] and the traffic can be classified as:

- Uniform traffic
- Diagonal traffic
- Log-Diagonal traffic

For N x N channels a matrix can be computed with input queue i and output queue j, such that the matrix output is

$A_{ij(n)} = 1$

And the expected value of the arrival rate matrix is

$\lambda_{ij} = E[A_{ij(n)}]$

For a finite system of FPGA controllers, the arrivals and departures are denoted as

$\sum_i \lambda_{ij} < 1$ And $\sum_j \lambda_{ij} < 1$

The sum of every row and column is less than equal to 1. The queuing analysis conducted in the preceding section analyzed packets of different sizes; whereas, scheduling within the FPGA fabric is queuing of fixed size cells for input and output queue at the FPGA buffer. The FPGA speed up depends on the reading speed from the input queues and

writing speed to output queues. The main objective function for FPGA controller is to maximize the read write process, S.

S = max(Sin, Sout)

The FPGA schedulers optimization depends on, the type of traffic and the algorithm used. Assuming Bernoulli arrivals, uniform traffic scheduling is the easiest to implement. The diagonal traffic is most common traffic and critical to schedule as compared to the uniform traffic. For fixed cell input and output queues there are two queuing strategies i.e. the input queue and the output queue. The FPGA controller has one queue on every input port. Each queue cells are served in as first come first served (FCFS) basis. In case of contention or non-availability of channel for input, the head of line (HOL) [25] blocking limits the throughput of the input queue. The output queue eliminates the HOL problem but its implementation is costly. The feasible solution is implementing a virtual round robin queue switching mechanism at the input port that can re-direct the input cells to matching ports without blocking. Virtual output queue (VOQ) is twice efficient than input queues [26]. FPGA controllers are less susceptible to large delay bounds and bursty non-uniform traffic. Therefore it is easy to model and simulate the incoming traffic delays. Non-uniform traffic rate matrix decomposition is governed by doubly stochastic matrices.

Decomposition of non uniform traffic rate matrices are solved by specialized algorithms such as Von Neumann (1953), Birkhoff (1946) and weighted fair queuing (WFQ) algorithms. The Von Neumann algorithm is a decomposition of rate matrix into doubly stochastic matrix. The doubly stochastic matrix is one that satisfies traffic rate matrix

conditions for both input and output channel the total traffic rate should be equal to 1 and less than equal to 1. Mathematically the rate matrix for input i and output j can be defined as

$\sum_i R_{ij} < 1$ And $\sum_j R_{ij} < 1$

The Birkhoff algorithm is decomposition of doubly stochastic matrix into combinatorial matrices which lies convex on the delay bound. The WFQ algorithm was designed at MIT and it uses tokens for the traffic rate matrix. The tokens are issued to the matrix in increasing order, with each token bearing a finishing time for every permutation matrix. The computational complexity of Von Neumann is O (N^4), Birkhoff algorithm is O ($N^{4.5}$) and WQF is O (N^3logN). FPGA controllers are frequently used in chemical and nuclear industry and network traffic across these FPGA controller is uniform; therefore, maximum weight matching (MWM) algorithm bearing a computational complexity of O (N^3) is a suitable option for an average weighted queue. MWM controller among all possible N iteration matching selects the VOQ with the highest weight i.e. the sum of edge matrix. FPGA array in industry has to schedule uniform but unknown traffic at moderate speed; therefore, a variation of MWM, greedy MWM with iterations can be implemented by using the longest queue and the oldest cell first terminology. It is a converging algorithm prioritizing the longest cells at VOQ.

For uniform traffic, the delay bound for (N x N switch with Q packets) MWM by Lyapunov assumption [27] is (ρ is link utilization factor)

$E[\|Q_{(t)}\|] \leq N^2 \frac{\rho}{1-\rho}(1 - \frac{\rho}{N})$

This researcher developed a variation of the delay bound for (N x N switch with Q packets) for iterative greedy MWM by Lyapunov assumption

$$\lim_{i\to\infty} E[\|Q_{(t)}\|] \leq \frac{N\Lambda}{\rho} + \frac{NB_{(b)}}{2\rho}$$

Where *Bb* is the weight difference to the MWM matching and the link utilization is given as

$$\rho = (1 - max \sum_{ij} \lambda_{ij})$$

And the permutation matrix is given as

$$\Lambda = \sum_{ij}(\lambda_{ij} - \lambda_{ij}^2)$$

The greedy algorithm of O (log K) complexity [28] is compared with the MWM algorithm at the input queued FPGA controller for a 2x2 FPGA fabric. The study shows that MWM algorithm of lower computational complexity is more efficient than greedy algorithm. In this simulation in Chapter six carried out a comparative analysis between the parallel iterative matching (PIM) which is a sub-maximal matching algorithm. In PIM, each edge of permutation matrix is added one at a time, and not removed from the matching until the end of iteration. The PIM is better in non predictive load as is the case with the internet traffic but here a predictive load is used for this case study of CANDU SCWR Hydrogen co-generation model. In such a case the iterative Greedy MWM average delays are found to be much better than iterative PIM algorithm as shown in Fig. 6.21 and 6.22.

4 DFM ANALYSIS

Previously, various methods were used to evaluate the control system reliability. These methods are identified in Chapter one of this research. These methodologies were developed for analog system with little or no evaluation for software decision support and control systems. The main draw backs to these systems are the scalability and static representation of the reliability model. Dynamic Flow Graph Methodology (DFM) is a well known reliability analysis method that can represent entire systems, and can generate a dynamic model for both the software and hardware conditions. In this chapter, both the fault tree and DFM model for the co-generation case study were used.

4.1 Probablistic Risk Analysis (PRA)

The probabilistic analysis is a quantitative analysis of the risk factor in mission critical systems used in various industries such as aerospace, chemical, and nuclear. There has been an extensive preventive analysis conducted by US Nuclear Regulatory Commission (NRC) to analyze data and prevent any probability of accident. The fault tree handbook [29] and PRA procedures guide by NRC in 1983 are examples that standardize risk assessment methodologies. Modern digital control systems analysis can be complex as system reliability analysis involves both software and hardware errors.

4.2 Dynamic Flow graph Methodology (DFM) Modeling

A DFM model is a cause and effect analysis of a system object with all possible combinations in the form of a directed graph. Fault tree models were based on binary input and could not satisfy the entire system state that comprises multiple timed variables and multiple binary logic conditions. The DFM modelling employs Multiple Value Logic

(MVL), which can represent the entire system and can generate a probabilistic risk assessment method for all system failure states. The model integrates the hardware and software with conditional operators such as AND, OR, XOR. The components of DFM are discrete or continuous variables, conditional branches, transition sequences merging various discrete values to produce top events in the form of prime implicants. The top event shows the state that can be critical for a particular system. The DFM method can analyse a system through deductive and inductive algorithms [30]. The deductive procedure is developed on a particular top event, normally taken as a failure state. This analysis is often termed as backward analysis from effects to causes. The backward analysis finds the set of variables through nodes, edges, transfer and transition boxes that generate the top event. The output of deductive analysis is a set of prime implicants. The deductive analysis is analogous to a fault tree analysis.

The inductive algorithm is often called the forward cause and effect analysis. For a particular set of inputs, the outputs can be normal outputs or may contain errors. The desired states can verify the system requirements whereas the undesired states can verify the system safety behaviour.

4.2.1 Process Variable Nodes

These nodes represent the physical variables and corresponding control system states. Digital control system processes can be represented by a software or hardware node. The node can be discrete or continuous with multiple states that can be generated according to the particular variable. A node can be failure or working, true or false, normal, low or high, or multiple conditions.

4.2.2 Causality Edges

The cause and effect relationship between different variables within the DFM model is defined through causality edges. The edge's functional relationship is developed by connecting to a transition box, or to a transfer box with the process variable nodes.

4.2.3 Transfer Boxes and Associated Decision Tables

The transfer box can represent the outcome of two or more process variables. The system knowledge can generate a logical relationship between the anticipated outputs. The possible combinations that can reflect the system's behaviour can develop the decision table. This decision table is based on logical operators and the number of inputs associated with it. The Pseudo code forms a logical link between various possible system states and predicts system behaviour in normal and abnormal conditions. The decision table entries are crucial for the system model, as improper selection can result in erroneous outcomes. A judicial combination results in a stable system model.

4.2.4 Condition Edge

Condition Edges are used to represent the control logic mapping input variable to output variable states. The conditional connection sets possible logical conditions for the associated transfer box. The set of inputs together with specific conditions associated with the transfer box can produce specific outputs.

4.2.5 Condition Nodes

The nodes represent the physical or software parameters. The conditions can be failure of component, process changes or software switching mode. The conditions are discrete variables with finite possible outputs.

4.3 Communication Network DFM Model

This report presents a general communication network model that can be used for different unit's group controllers. There are six levels of control for the DFM model; Hardware control (Two: One each for pre and post processing), OSI layer control (Two: One each for pre and post processing), Protocol and data control, and transmission control. The control strategy used in this study is a combination of both hardware, software control through appropriate protocol, communication medium and transmission algorithm.

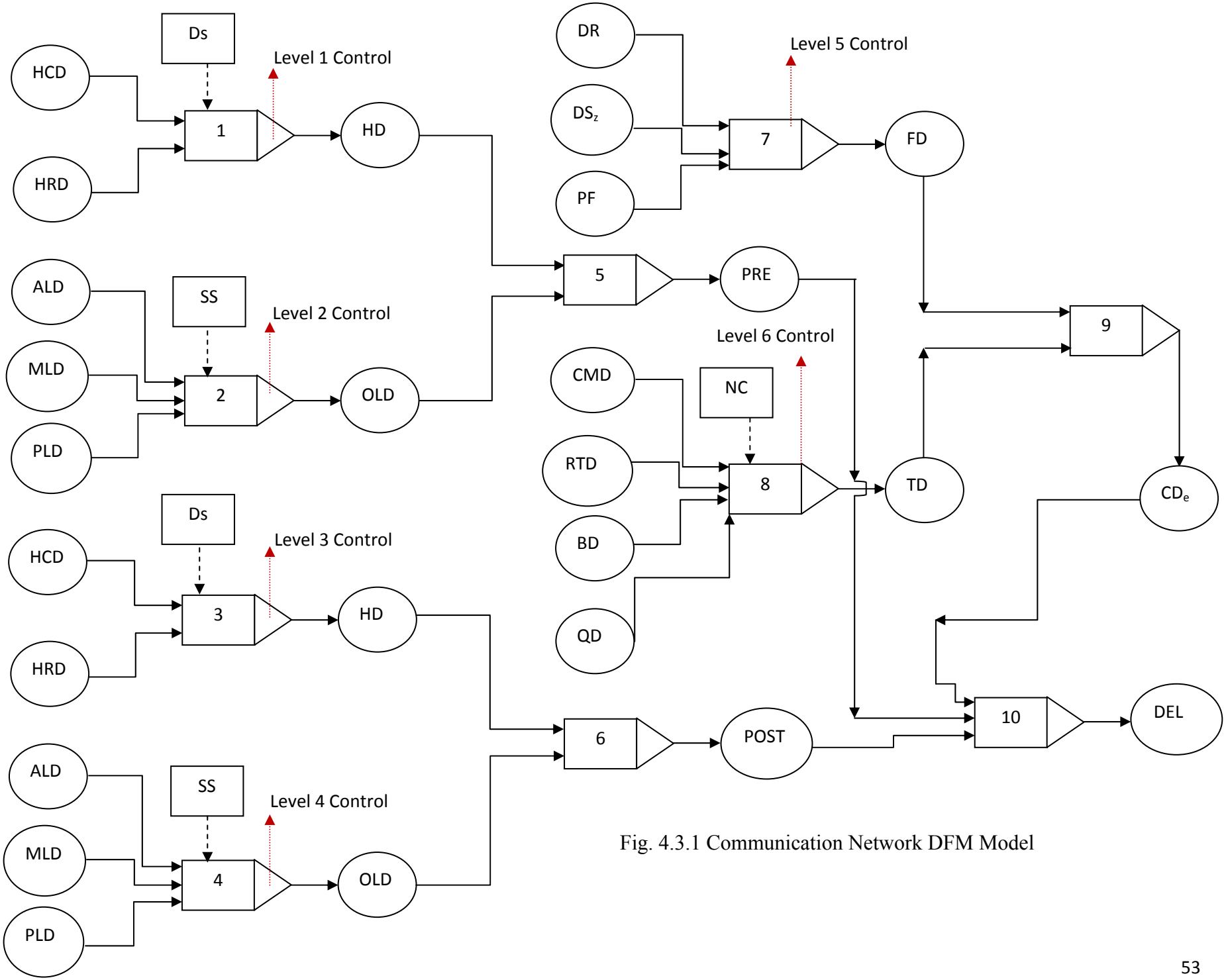


Fig. 4.3.1 Communication Network DFM Model

Fig. 4.3.1 shows the six levels of controls for the communication network. Level 1 control is Hardware Delay (HD) which comprises Hardware Computational Delay (HCD) and Hardware Response time Delay (HRD) Ds is the Device status.

Level 2 control is the OSI layer delay which comprises Application Layer Delay (ALD), Mac Layer Delay (MLD) and Physical Layer Delay (PLD), SS is the software status.

Both Level 1 and 2 are pre-processing delay.

Level 3 and Level 4 are the same controls as Level 1 and Level 2 but for post-processing delay.

Level 5 is the Frame Delay (FD), which comprises the Data Rate (DR), the Data Size (DS_z), and the Protocol Frame (PF).

Level 6 is Transmission Delay (TD), which comprises of Communication Medium Delay (CMD), Retransmission Delay (RTD), Queuing Delay (QD), and Back off Timer Delay (BD). NC is the network condition; either it is contention free or contention full.

Transmission Delay (TD) and Frame Delay (FD) together form the Communication Delay (CD).

Pre-Processing Delay, Post-Processing Delay and Communication Delay combined together form the total Delay (DEL).

4.4 Comparative Model

For this case study of CANDU SCWR Hydrogen co-generation cycle the fault tree and DFM model methodologies were used. The results indicate that fault tree has drawbacks as compared to the DFM model. Fault tree can only represent a part of the system state, with limited or no scalability option. It does not represent the system's changing behaviour for software, hardware and network condition states. All of the system conditions can be combined with components discrete conditions that can generate results for multiple combinations of OR, AND, and XOR logics. DFM also represents system intermediate conditions in the form of intermediate node top events and final node top events. Fault tree only represents system hardware logics with fixed states as inputs and outputs.

5. CANDU SCWR HYDROGEN CASE STUDY

5.1 Super Critical Water Reactor (SCWR) System

Current SCWR research and development focuses on two different technologies. The pressure tube CANada Deuterium Uranium (CANDU) reactor design and the US pressure vessel Light Water Reactor (LWR) reactor design. In this model the CANDU Super Critical Water Reactor (SCWR) technology is assumed. The main features of CANDU SCWR system are higher thermal efficiency (45 to 50%) and simplified system configuration due to one phase superheated steam cycle that contributes to lower cost by the elimination of certain equipments and thereby less expensive electricity. The CANDU SCWR primary heat transport system uses light water that operates at a temperature and pressure higher than 600^{o}C and 22.1MPa [31-35].

Super critical water fossil fuel power plants have been operating for a long time and the operating experience and design features can be adapted in the CANDU SCWR nuclear reactors. CANDU operational knowledge forms the basis of generation III and generation IV advance technologies including the Advanced CANDU Reactor (ACR) and the Super Critical Water Reactor (SCWR). The CANDU SCWR is an advanced reactor based on technologies such as the Pressurized light Water Reactor (PWR) and the Boiling Water Reactor (BWR) designs [36], operating at a high temperature and pressure with opportunities of improved efficiency and reduced operational cost due to high thermodynamic single phase heat transport medium.

In recent years, significant efforts are underway to develop the hydrogen co-generation model by CANDU SCWR associated enabling technologies. The CANDU SCWR heat and steam is planned to be used for hydrogen production by copper chloride method which is increasingly viewed as a strong component of zero emission energy technologies. The SCWR is still in its evolution phase; therefore, collaborative efforts are required to better understand SCWR plant materials, chemistry, safety, reliability, and stability methods. It is anticipated that SCWRs would have a combination of advantages of BWRs and PWRs [37].

The CANDU SCWR reactor design is based on two concepts: The conventional light water reactors (PWR) and the heavy water reactors (CANDU). The super critical steam generators and turbine design in conventional power plants spans over fifty years [37]. This experience of super critical turbines can be utilized effectively in CANDU SCWR systems. Previous studies focused on conventional SCWR cycle for fossil fuel power plants but recently there has been a growing interest for the CANDU SCWR cycle [38]. Since CANDU SCWR concept is still in its evolution phase several optimal configurations such as direct heat cycle, indirect heat cycle and dual reheat steam cycle can be investigated to form an efficient design.

5.1.1 Direct Reheat Cycle

Expanding the reactor super critical steam directly to the high pressure turbine is a more efficient method as temperature is contained within the cycle and replaces the use of expensive steam heat exchangers, dryers and moisture separators but a

high pressure turbine is directly exposed to the reactor steam, which can be a source of radioactive spread in the conventional cycle. Fig. 5.1.1 shows the direct steam CANDU-SCWR cycle, with a moisture separator between the intermediate and low pressure turbine.

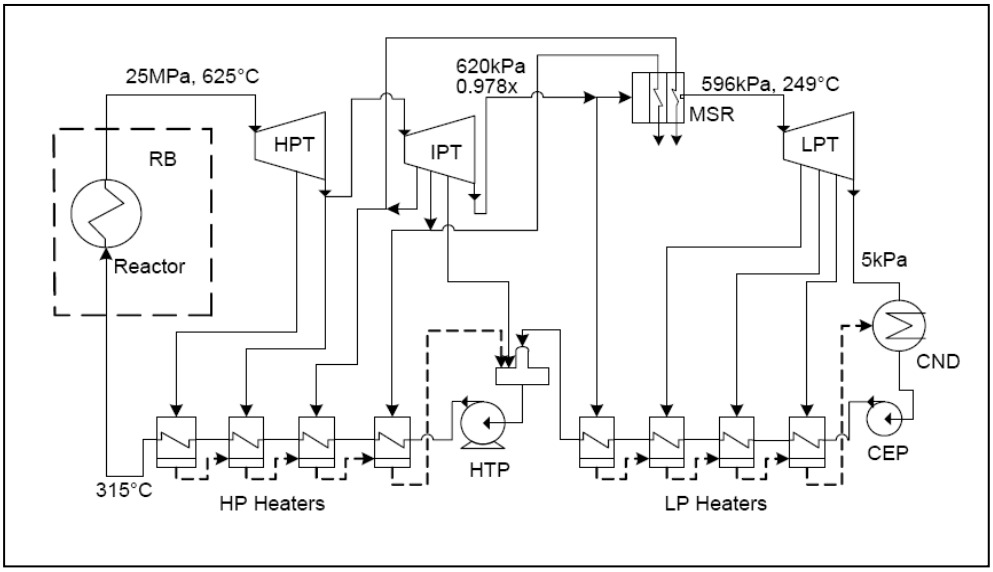


Fig 5.1.1 CANDU-SCWR Direct Steam Cycle [Ref.: 39]

5.1.2 Indirect Cycle

CANDU systems use the indirect method of steam expansion from reactor to the intermediate heat exchangers, dryers and separators.

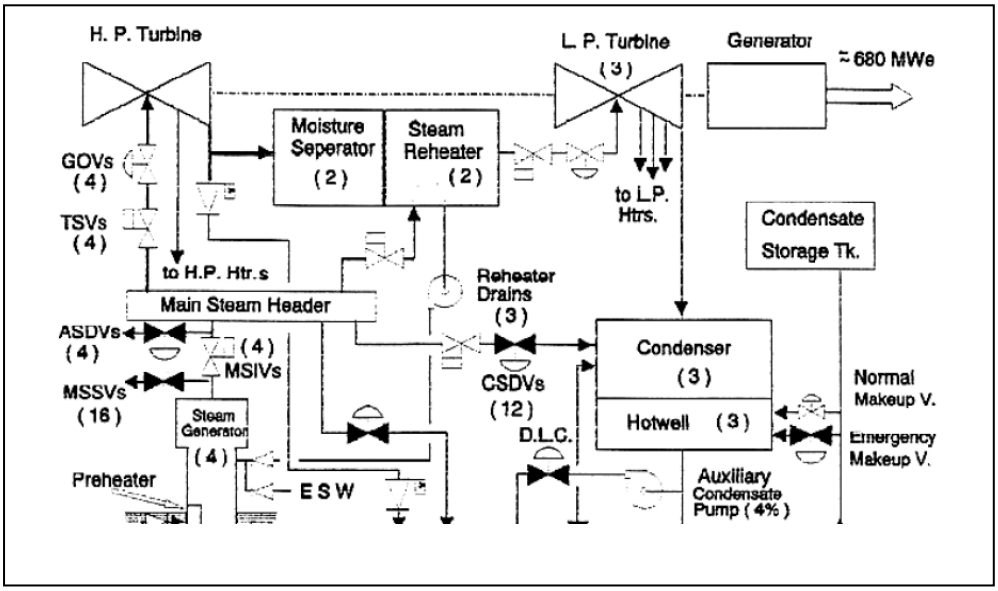


Fig 5.1.2 CANDU Indirect Steam Cycle [40]

Thermal efficiency is sacrificed in the dual expansion cycle but radiation containment is achieved through dual expansion cycle. Fig. 5.1.2 shows the steam expansion into moisture separator and re heaters.

5.1.3 Dual Reheat Steam Cycle

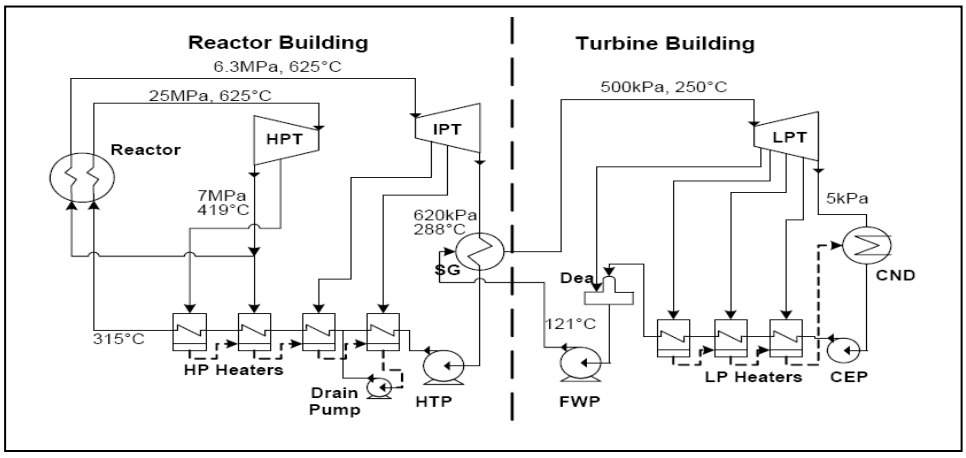


Fig 5.1.3 CANDU-SCWR Dual Reheat Steam Cycle [Ref.: 39]

Figure 5.1.3 shows the regenerative loop of a dual reheat steam cycle for both the reactor and the feed water system. The parametric ranges achieved through regenerative loop can be up to 625 degrees centigrade and 6.3 MPa. These are ideal for the CANDU SCWR hydrogen co-generation model.

CANDU SCWR cycle efficiency is improved by employing a regenerative heat mechanism within the SCWR reactor and also into the feed water system. The SCWR heat regeneration is accomplished by recirculation cycle between the HP turbine and the SCWR, which can achieve higher temperature steams at lower pressures. It is suitable for CANDU SCWR hydrogen co-generation model. Steam bleed from high pressure and low pressure turbines is led to feed water system heat exchangers to increase the feed water temperature up to 315 degrees centigrade. Comparative analyses between no-reheat, single reheat and double reheat reveals that the single reheat CANDU SCWR cycle is optimal considering the design complexity and thermal efficiency of the cycle. The major problem for integrating the hydrogen cycle into the CANDU SCWR cycle is the higher pressures of the SCWR heat transport cycle. The higher pressures are necessary to maintain the steam in the super critical state. The pressure ranges in SCWR with non-regenerative design is 25 MPa and 625 degrees centigrade while it is from 9 to 6 MPa and 625 degrees centigrade in the regenerative loop. In copper chloride cycle, the oxygen production, molten copper chloride unit requires 500 degrees centigrade while hydro-chloric cycle requires 430 degrees centigrade. A lower pressure differential between the integration loops is essential to meet the safety requirements of both the CANDU SCWR and the hydrogen plant.

5.2 COPPER CHLORIDE HYDROGEN PRODUCTION UNIT

The copper chloride electrochemical hydrogen production cycle is divided into five major steps.

- Chlorination Step
- Electrolysis Step
- Drying Step
- Hydrolysis Step
- Decomposition Step

The heat generated by copper chloride components can meet fifty percent of a plant's heat requirements. The chlorination, hydrogen production exothermic step occurs at temperatures 450-470 ◦C [41] producing molten copper chloride and hydrogen gas. The second step of electrolysis is accomplished in an aqueous solution of HCl with temperatures between 30-80◦C. The third, endothermic drying step can be accomplished by crystallization with temperature ranges from 30-80◦C or by spray drying with temperature ranges from 100-260◦C. The fourth endothermic hydrolysis step is accomplished at 375◦C. The final step involves endothermic decomposition producing oxygen at 530◦C. The chlorination and decomposition steps require higher operating temperatures from 470 to 530◦C. Therefore, the CANDU SCWR high temperature and low pressure loop can be interfaced with chlorination and decomposition units.

Fig. 5.2.1 shows one line diagram for major components of a close loop hydrolysis reactor. In hydrogen production the copper takes many forms such as solid, slurry and molten. In the hydrogen production unit (ion exchange bed) solid copper reacts with high temperature hydrochloric gas to form hydrogen and copper chloride. The hydrogen gas produced has temperatures from 430 to 475◦C, which is further cooled down to 25◦C by the cooling water and is passed through the purifier before storing it into the hydrogen storage tank. Molten copper chloride produced in an ion exchange bed is spray dried to solid copper chloride and slurried with hydrochloric and water solution. The slurry is then pumped to the conveyer belt for the repetitive cycle. The cooling water exchanging heat with hydrogen gas enters heat exchanger at 25◦C and leaves it at 400◦C.

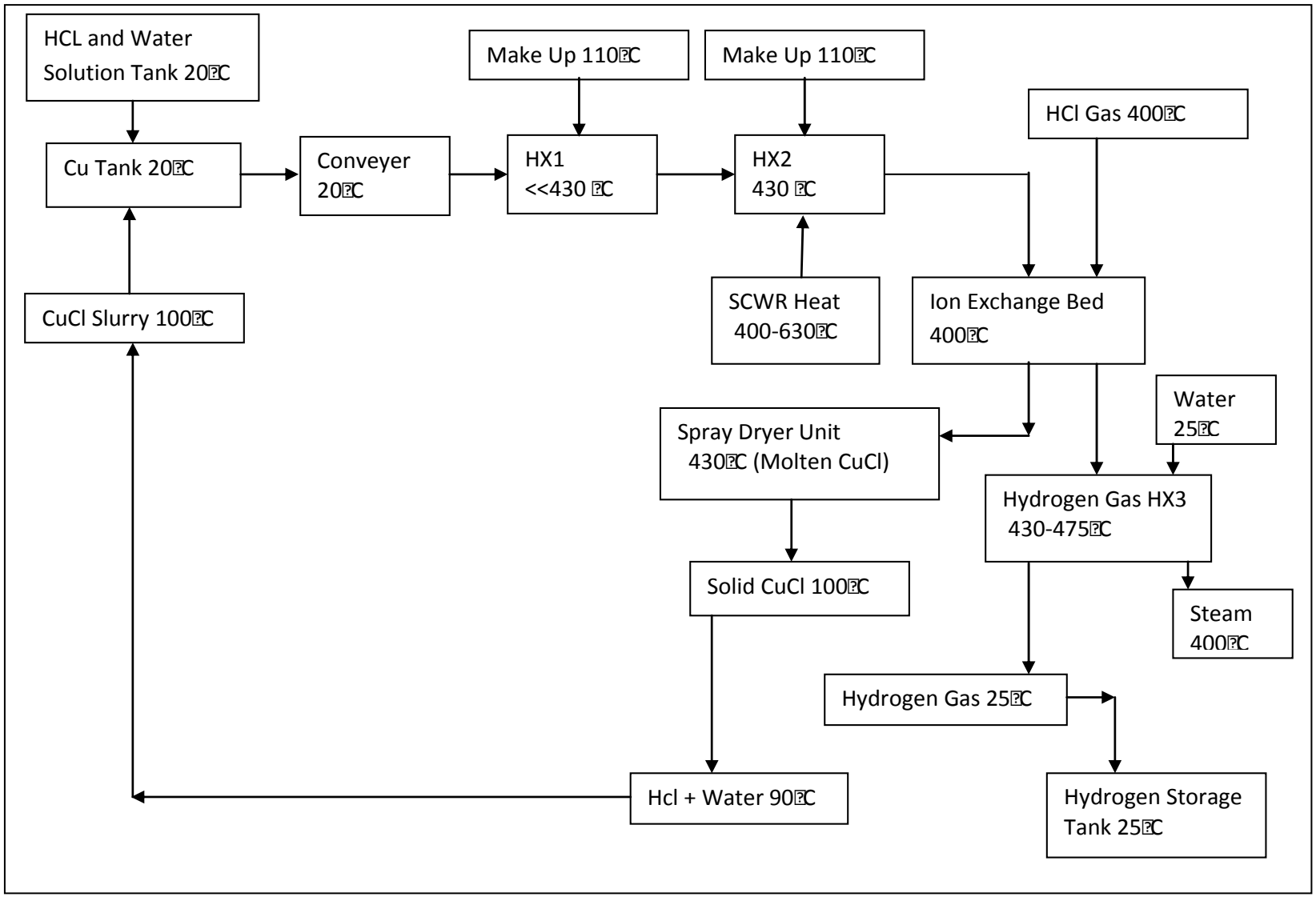


Fig 5.2.1 Hydrolysis Reactor Loop

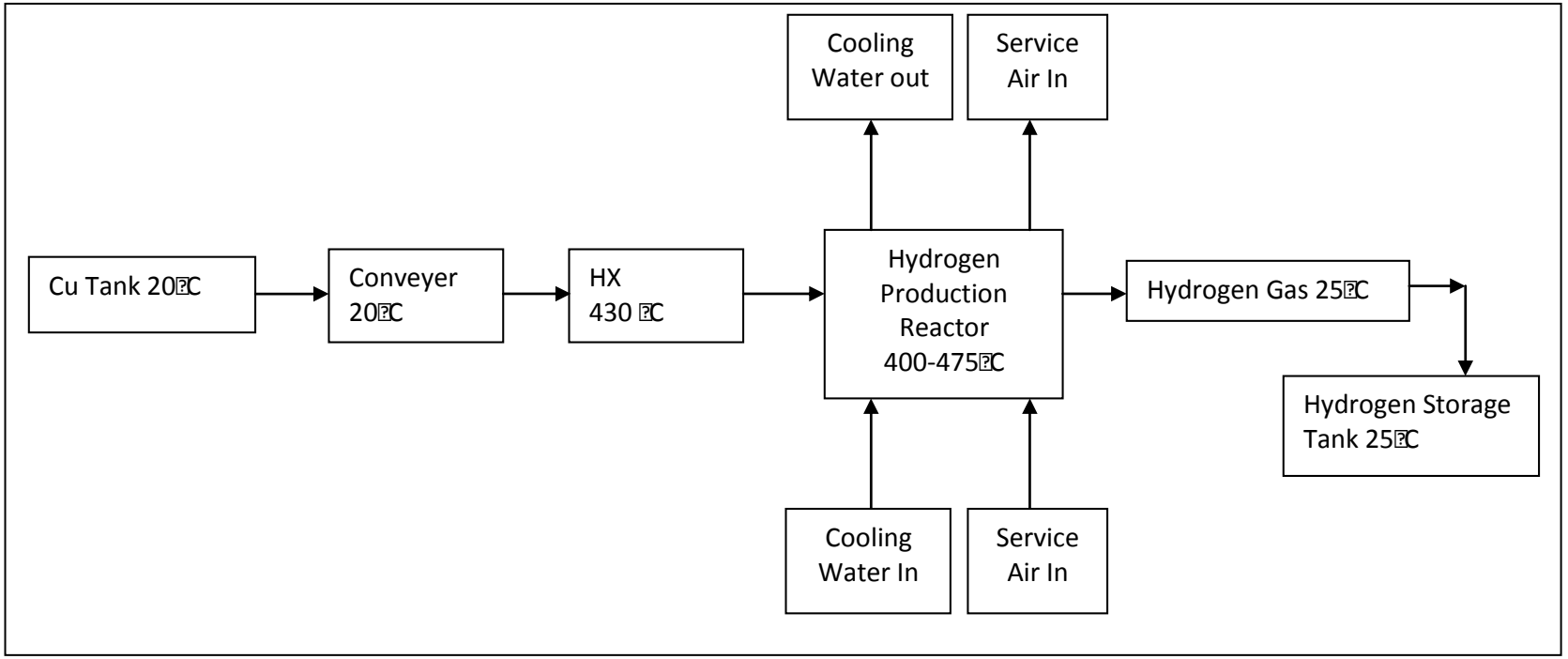


Fig 5.2.2 Simplified Hydrolysis Reactor Loop

The simplified version shows the coolant cycle and instrument air supply for hydrogen unit.

Fig. 5.2.2 shows a simplified hydrolysis reactor loop, with copper going in and hydrogen coming out. The water and service air supply is essential for the heat exchange in hydrogen reactor unit.

P&ID Components of Hydrolysis Reactor Loop	
HW-TK1	HCl + Water Tank
CU-TK2	Copper Storage Tank
HS-TK4	Hydrogen Storage Tank
CCS-TK3	Copper Chloride Slurry Tank
CU-CN1	Copper Conveyer Belt
A1	Fluidized Bed
A0	Ion Exchange Bed
A2	Coolant Water Supply

Table 5.2.1 Acronyms of P&ID Controller of Hydrolysis Reactor Loop

Fig. 5.2.3 shows the hydrolysis loop of the hydrolysis unit with supply lines, valves, non-return valves, flow-rater, temperature controllers and pumps. The P&ID shows Lines 1-1 to 1-8, Flow controller 1.11 to 1.64, Temperature monitors

from 1.41 to 1.72, Pumps from PM1 to PM8, valves from V1 to V9, Non-return valves from NV1 to NV9.

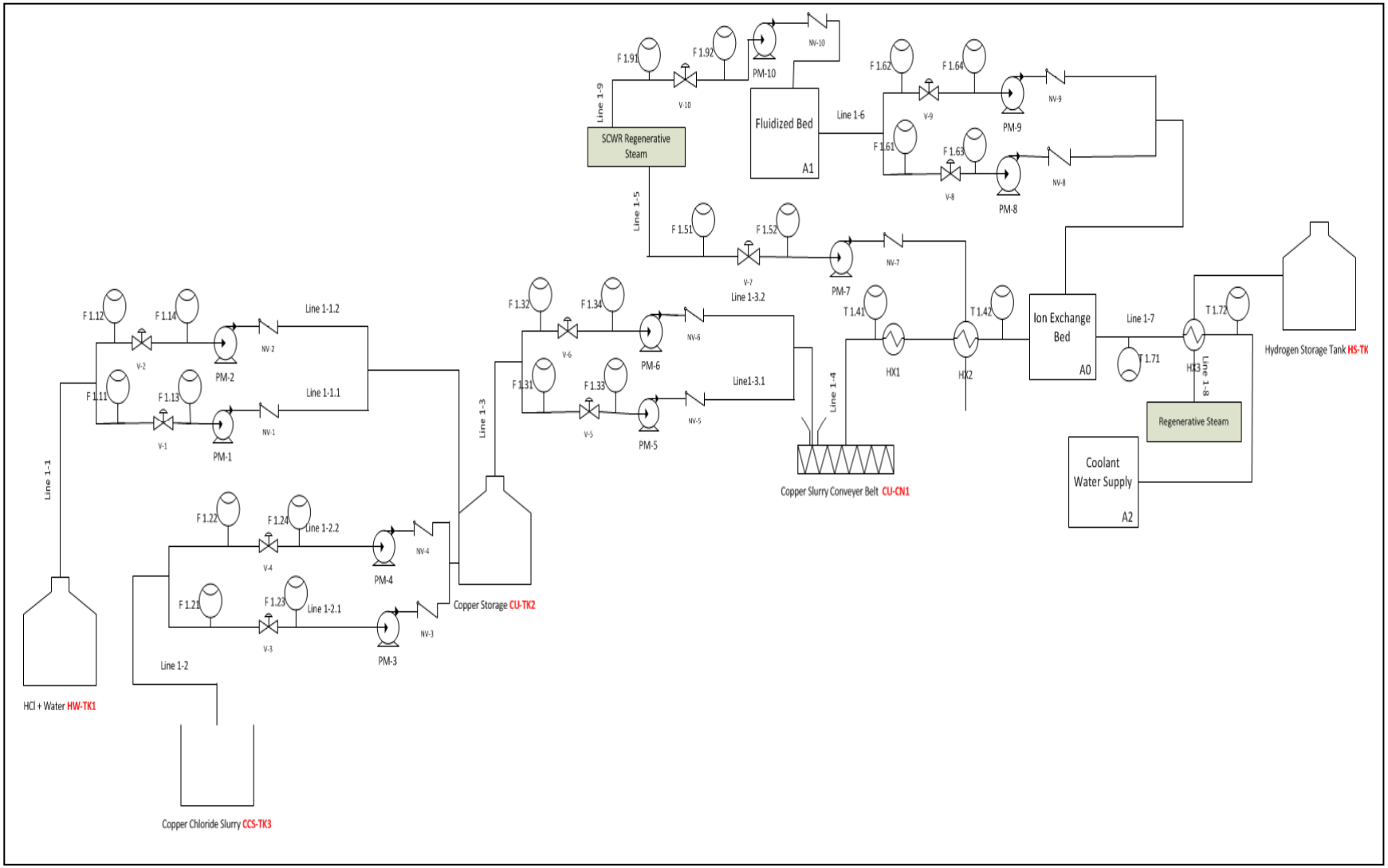


Fig. 5.2.3 P&ID of Hydrolysis Reactor Loop

Fig. 5.2.4 shows the single line close loop diagram of oxygen reactor unit. Oxygen is produced from cupric oxide copper chloride slurry ($CuO*CuCl_2$), Cupric oxide copper chloride slurry is heated to decompose into copper chloride and oxygen at 500◦C. The decomposition heat is provided by SCWR steam cycle. Oxygen at 500◦C is cooled down to 25◦C through coolant and stored in oxygen tank. The byproduct of reaction, molten copper chloride is cooled down up to 100◦C in spray dryer unit, and slurried with water and hydrochloric acid solution and stored in copper storage tank. Fig. 5.2.5 shows a simplified oxygen production unit one line diagram. The main units are oxygen reactor unit, spray dryer unit, sedimentation cell, and oxygen reactor. A continuous supply of water and service air is essential for the reaction.

Fig. 5.2.6 shows P&ID of the oxygen reactor. Copper storage tank supplies copper to the sedimentation tank where it is mixed with hydrochloric acid and water solution. This slurry is forwarded to the fluidized bed. The copper chloride slurry is decomposed into hydrochloric gas and cupric oxide copper chloride.

P&ID Components of Oxygen Reactor Loop	
CS-TK4	Sedimentation Cell
CU-TK2	Copper Storage Tank
OS-TK5	Oxygen Storage Tank
A2	Coolant Water Supply
A3	Oxygen Unit

A1	Fluidized Bed

Table 5.2.2.Acronyms of P&ID Controller of Oxygen Reactor Loop

The P&ID shows Lines 1-8 to 1-11, Flow controller 18.1 to 1.10.4, Temperature monitors from 1.8.1 to 1.11.2, Pumps from PM10 to PM14, valves from V11 to V13, Non-return valves from NV11 to NV13.

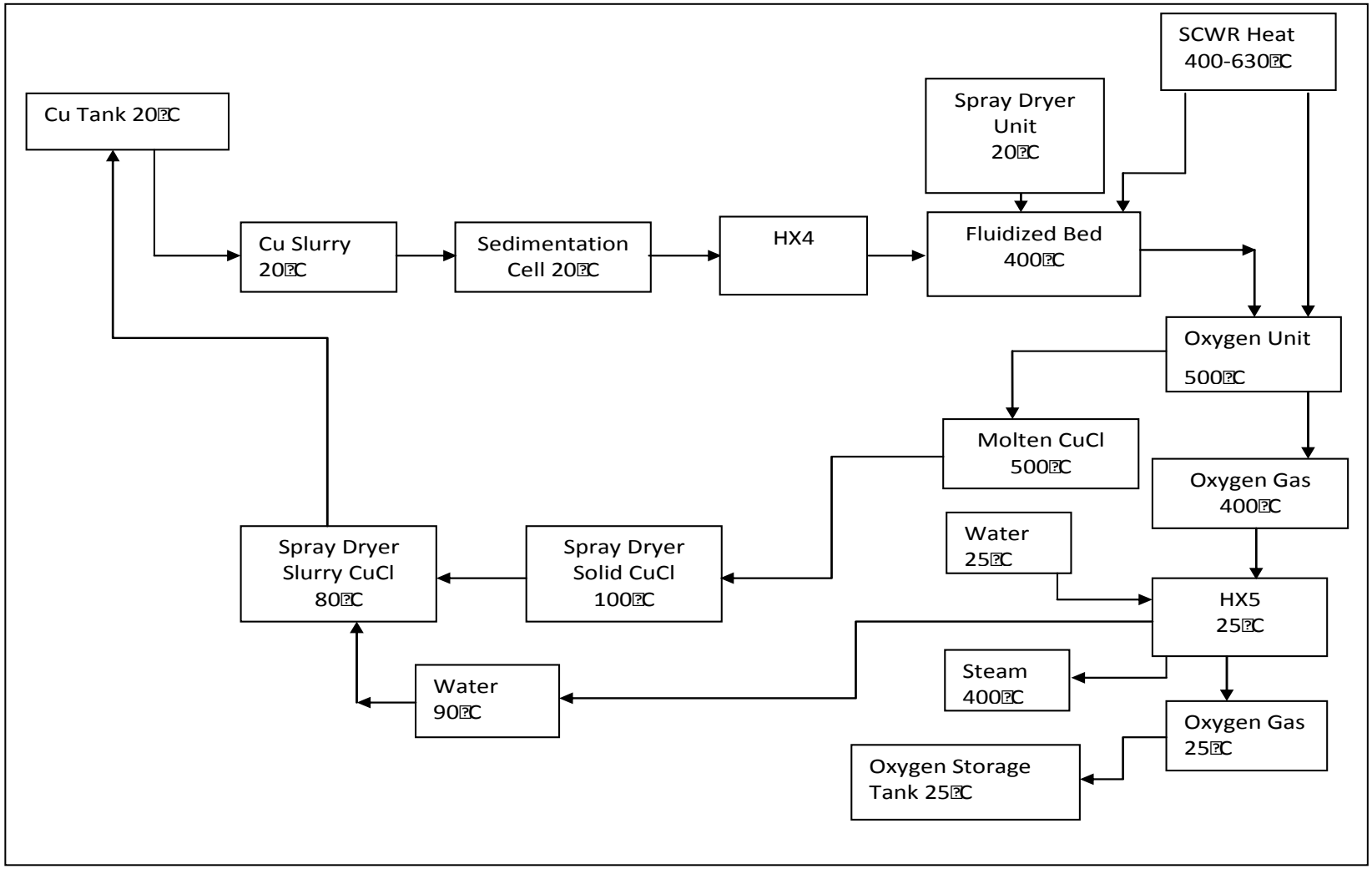


Fig 5.2.4 Oxygen Reactor Loop

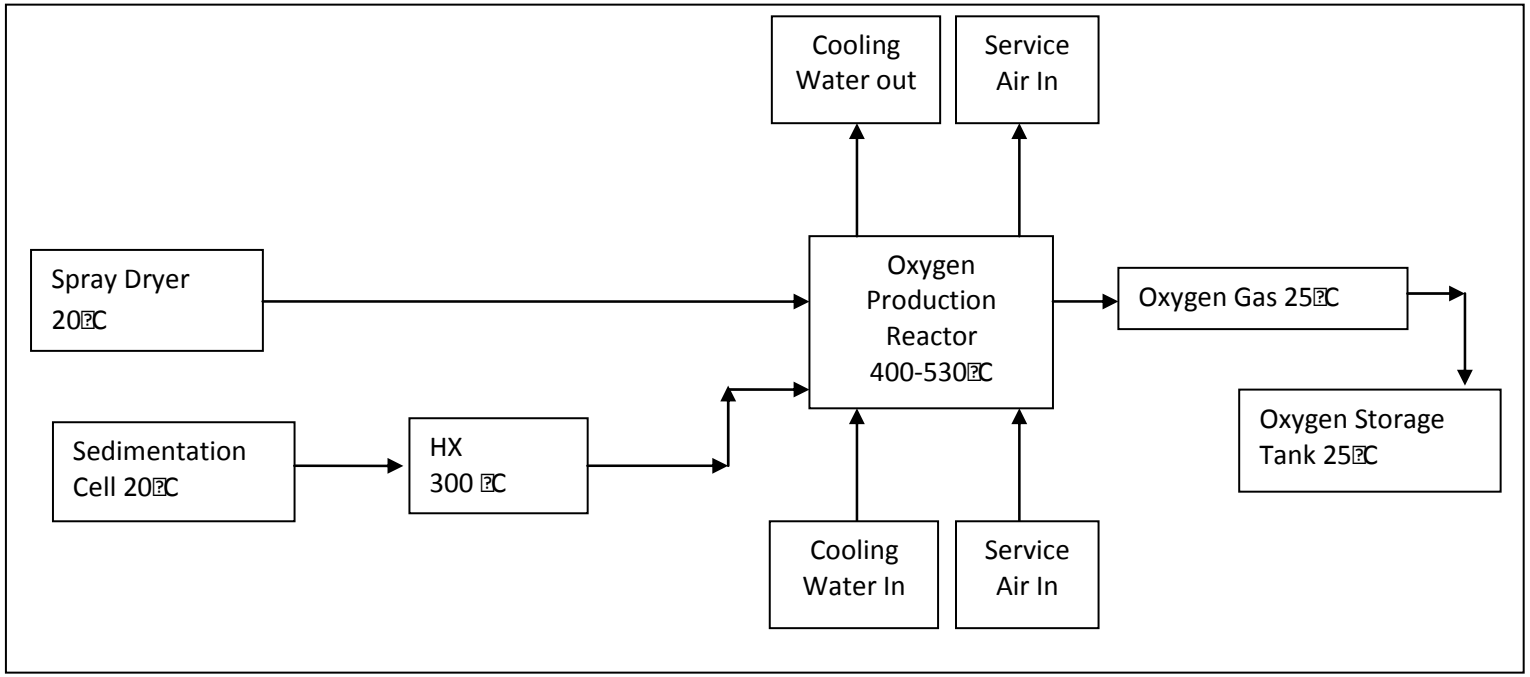


Fig 5.2.5 Simplified Oxygen Reactor Loop

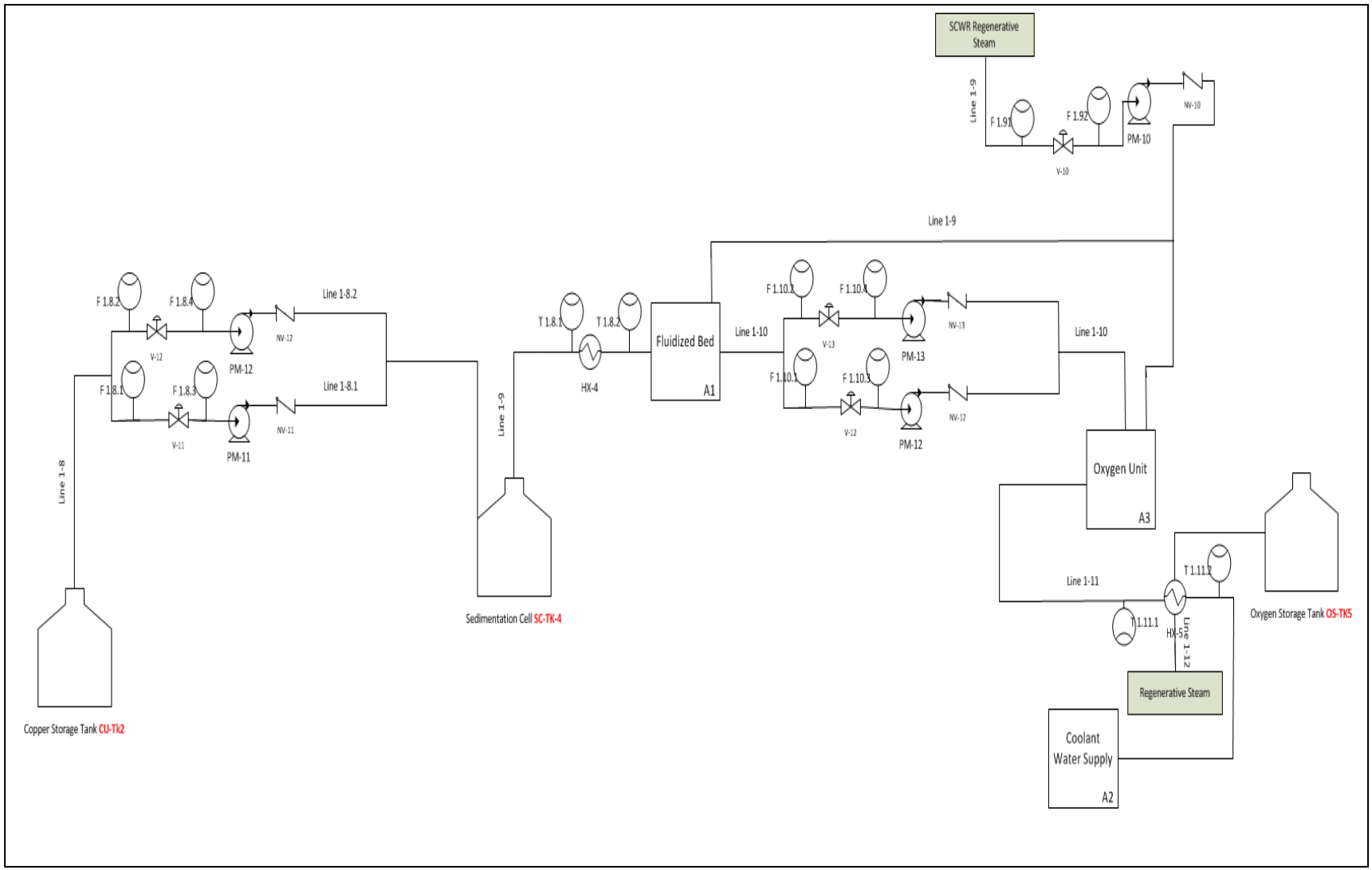


Fig.5.2.6. P&ID of Oxygen Reactor

5.2.1 Co-Generation Model

Various electrochemical processes [42-44] for hydrogen production are identified and currently under review but very few have been found to be feasible. The methods and cycles which are proven viable for hydrogen production contain Steam Methane Reforming (SMR), Sulphur-Iodine (S-I) and Copper Chloride (Cu-Cl) cycles. The SMR cycle operates at higher temperatures from 700 to 800 ◦C and it is not free from carbon emission; whereas, the copper chloride hydrogen cycle operates at lower temperature i.e. 600◦C, and is also free from carbon emission. Hydrogen production using direct electrolysis and combination of electro-chemical process can maintain the nuclear plant's efficiency by utilizing its wasted heat and is also a source of clean energy hydrogen fuel.

Table 5.2.1 shows the CANDU SCWR hydrogen outlet temperature which is in the ranges from 600 to 625◦C and with pressures ranging from 25 to 6 MPa. Two loops in CANDU SCWR reactor are assumed. Loop 1 with high temperature and pressure, loop2 with high temperature and low pressure. The loop 1 is the main heat cycle loop of the reactor and loop2 is the regenerative loop between HP turbine and SCWR. Both loop1 and loop2 heat can be utilized in the copper chloride hydrogen production unit at oxygen production unit (500◦C) and fluidized bed (400◦C).

Table 5.2.2 shows the five step copper chloride chemical reactions. The higher temperature reactions are a decomposition of cupric oxide copper chloride into oxygen gas, and hydrogen gas production. The CANDU SCWR heat is expanded

in these two loops. Oxygen gas is produced at high temperature; similarly, hydrogen gas is also produced at high temperatures and both are cooled through heat exchangers. The regenerative heat produced in these steps can be utilized within the closed loop cycle and can meet up to 40-50% of Cu Cl loop heat requirement.

Parameters	**Unit**	**CANDU-SCWR**	
Temperature (Reactor Core)	**^{o}C**	**Inlet**	**Outlet**
		350	**625**
Pressure	**MPa**	**25/6**	
Thermal Conductivity	**W/m-k**	**Inlet**	**Outlet**
		0.481	**0.107**
Temperature Increase (Inlet-outlet)	**^{o}C**	**275**	
Enthalpy Increase (Inlet-outlet)	**KJ/Kg**	**1943**	
Specific Heat	**J/Kg-K**	**Inlet**	**Outlet**
		6978	**2880**

Table 5.2.1 Thermo /physical Parameters of CANDU-SCWR

Reaction	Temp. ^{o}C
$2Cu(s) + 2HCl \longrightarrow CuCl(l) + H_2(g)$	430 – 475
$2Cu(s) + 2Cucl(Aq) \longrightarrow CuCl(Aq) + Cu(s)$	30 – 80
$CuCl_2(Aq) \longrightarrow CuCl_2(s)$	30- 80 (crystallization) 100-260 (Spray Drying)
$2CuCl_2(s) + H_2O(g) \longrightarrow CuO^* CuCl_2(s) + 2HCl(g)$	375- 400
$CuO^* CuCl_2(s) \longrightarrow 2CuCl(l) + 1/2O_2(g)$	500 -530

Table 5.2.2 Thermo /physical Parameters of Cu-Cl Reaction

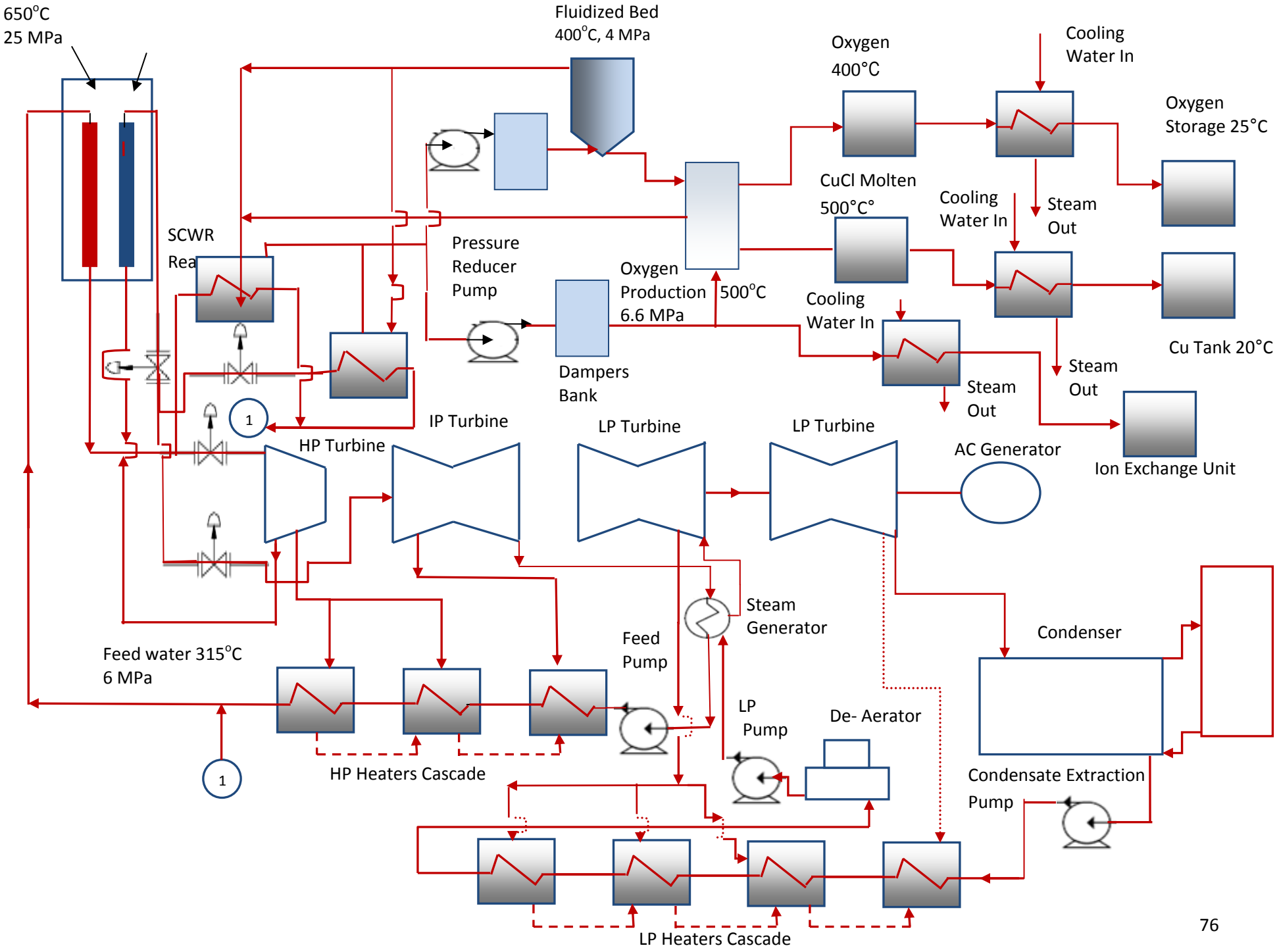


Fig 5.2.7 Schematic Diagram of Co-Generation of SCWR and CuCl Hydrogen Cycle

5.2.2 Co-Generation Strategy

The CANDU-SCWR design uses single/dual cycle, direct/indirect heating methods, but as thermal efficiency is higher in single cycle direct heating system, it is assumed that the CANDU-SCWR design as single cycle direct heating mode. The intermediate pressure turbine (IP) between the high pressure (HP) and low pressure (LP) turbines are omitted, assuming IP and LP turbines bearing the same efficiency. It was also assumed that multiple stage single shaft turbine-generator cascade design for later stage steam expansion. The LP turbines in this model are the double flow configuration turbine to handle the large amount of steam. This CANDU SCWR hydrogen cogeneration cycle is comprised of: a super critical reactor, a supercritical turbine, which consists of one High-Pressure (HP) cylinder, and multiple Low-Pressure (LP) cylinders, one de-aerator, seven feed water heaters, and pumps. A regenerative method for reheating the steam leaving condenser hot well through cascade LP heat exchangers getting the sub-critical heat from LP turbines bleed valves was also assumed in this study. The same regenerative heating mechanism is adapted at the HP heat exchangers to attain the reactor inlet temperature of 350 ^{o}C.

5.2.3 Load Variation Effect on CANDU SCWR Cycle

Load variation is a major cause of concern for both operation and safety standards in CANDU reactors. For reactor excessive steam generated during sudden load variation, the CANDU systems utilize steam rejection to condenser and atmosphere as a remedy; moreover, sudden load variations can be controlled by

the reactivity controls and turbine hydraulics (load following) but it is a system overhead which can be reduced through the induction of copper chloride hydrogen production unit as shown in Fig. 5.2.7 The copper chloride cycle requires comparatively moderate temperatures to produce hydrogen and CANDU SCWR reactor can provide heat to the Oxygen molten salt reactor (500 degrees centigrade), and to the fluidized bed (400 degrees centigrade). The advantages of hydrogen module in CANDU SCWR are improved efficiency in terms of heat utilization and increase in system reliability by accommodating the load variation. Our CANDU SCWR CuCl model utilizes the heat from CANDU SCWR reactor, to separate hydrogen and oxygen and recycle the CuCl by products during the production. The load variation is not only the wastage of resources, but it is also a safety concern due to excessive steam. The data is available online at www.theweathernetwork.com (accessed at May 2011) for the variation of the supply and demand of the power generated in Ontario region. We analyze the variation between specific time and formulate a relation between excessive steam available during off peak hours and CANDU SCWR hydrogen production. It can be seen from the graph in Fig 5.2.8 that the minimum energy requirement is between 2 to 6 am, with an exponential growth between 6 to 9 am, and is maximized between 4 pm to 8 pm. The extraction time for steam can be synchronized with the hydrogen production unit for better and efficient co-generation mechanism. Also, for weekends there are extra megawatts of power produced due to excessive steam.

These values were plotted for the energy used for power generation in percent of peak demand hour on y-axis and excess megawatt electrical and thermal available for hydrogen production on x-axis. Fig. 5.2.9 shows the plot. The reference standard as the peak demand hour from 6 pm to 8pm, represented as 100 on the y-axis, a high demand hour between 9 am to 4 pm which is represented as 92 on y-axis, and minimal demand hour between 1 am to 5 am represented as 66 on y-axis. The corresponding megawatts of electrical and thermal power, available for hydrogen production are plotted at x-axis. The maximum power available for hydrogen production is at minimal demand hour that is between 1am to 5am and the minimum power available for hydrogen production is between 6pm to 8pm which is the maximum demand period.

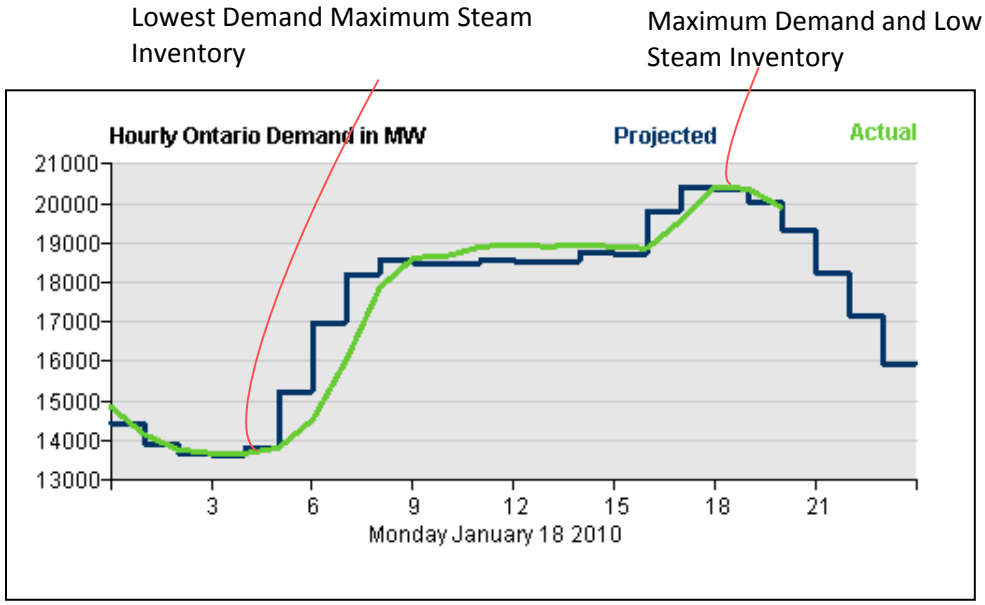


Fig. 5.2.8 Load Variation in Ontario for a single day

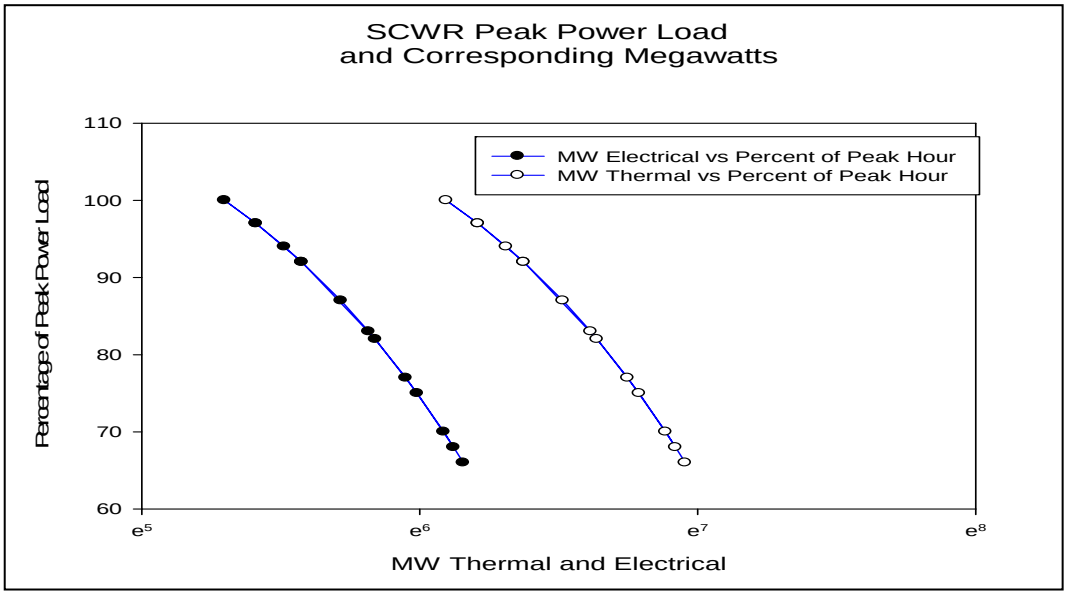


Fig. 5.2.9 MW Available for Hydrogen Production during peak and off peak Hours

The time cycle for the steam generation is assumed for a twenty four hour day, and the load variations are taken from the data available at the weather network website. All the power is assumed to be generated through SCWR plant with a full power capacity of 100 MWe and the copper chloride regenerative loop of exothermic reactions produces 50% of its own steam heat requirement.

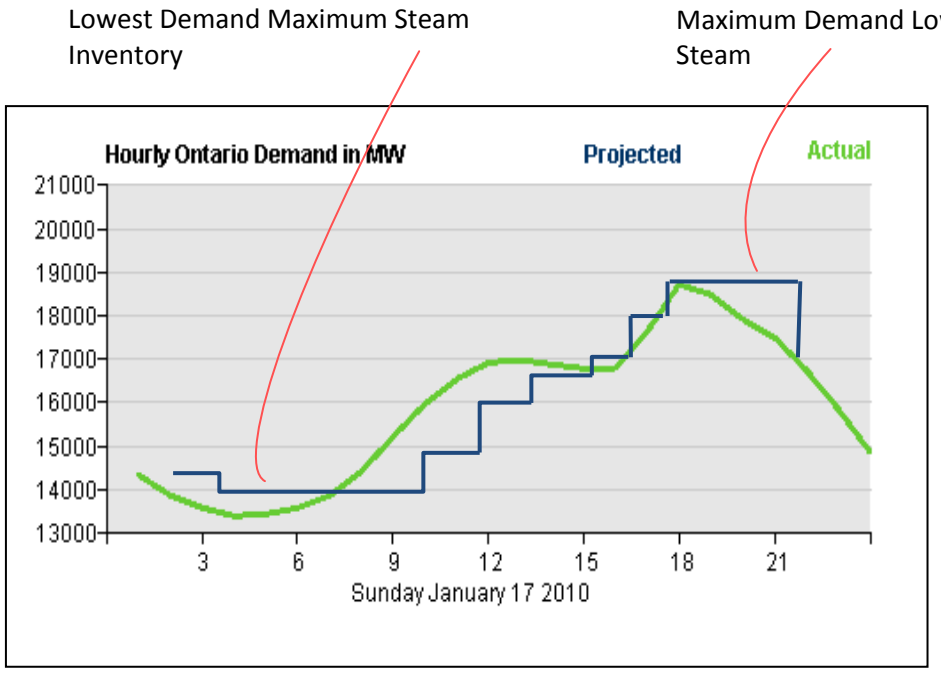


Fig. 5.2.10 Load Variation in Ontario for a week end day

Fig. 5.2.10 shows the load variation for a weekend day that is Sunday, January 17, 2010. Two well known methods normally used to cope with the excessive steam problems are turbine following reactor and reactor following turbine. Turbine following reactor mechanism is the activation of turbine hydraulics to relieve the steam to atmosphere in case of excessive steam. Reactor following turbine mechanism uses the control absorber rod reactivity control mechanism to control the chain reaction and hence control the excessive steam generated. These control actions can be eased through the induction of the hydrogen unit co-generation model.

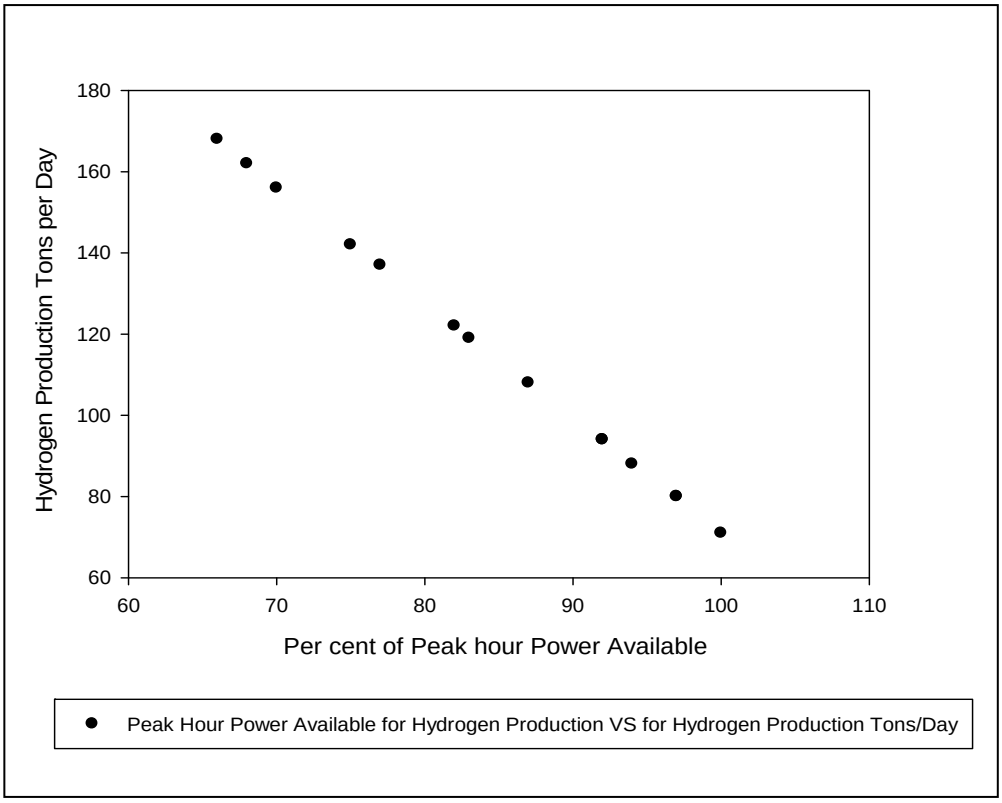


Fig. 5.2.11 CANDU-SCWR Hydrogen Co-generation Capacity

The computed value of the Hydrogen production for copper chloride is 162 Mega Joules per kilo gram of hydrogen with the assumption that 50 per cent of the energy is provided within by exothermic reactions in the copper chloride cycle. In Fig. 5.2.11, the x-axis represents the percent of peak hour ranging from 66 as the lowest and 100 as the highest. The maximum hydrogen production is 168 tons per day which is achieved between 1am and 5 am, the off demand hours and the minimum is 71 tons per day between 6pm to 8pm, which is the peak demand hours.

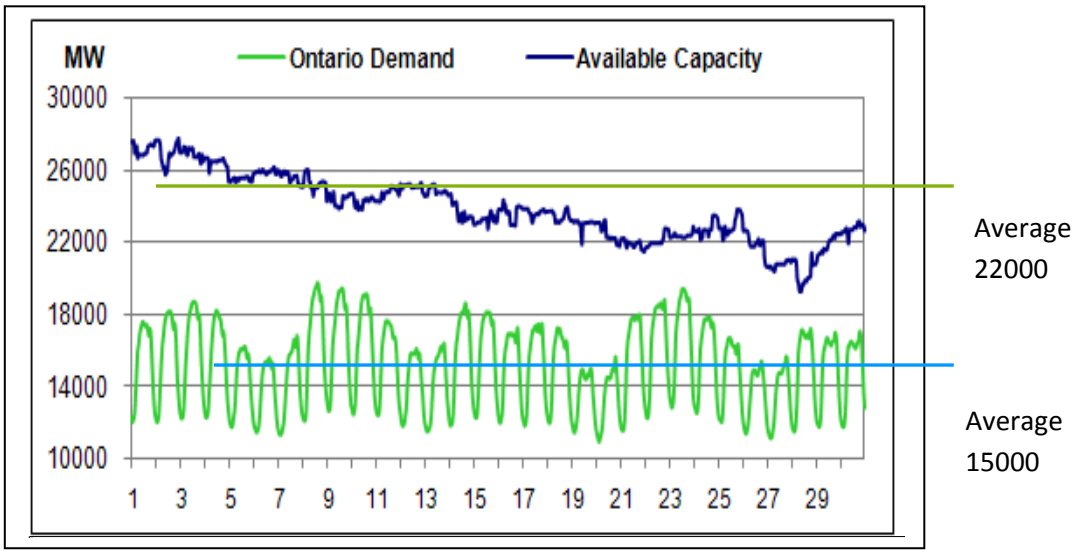


Fig. 5.2.12 Ontario Demand and Capacity Load Variation for a Month

The above statistics shows an excessive 7000 MW_e capacity that is not utilized and the heat generated in nuclear and conventional power plants is wasted. The 7000 MWe is about 32 per cent of the total production, which is wasted. The Co-generation hydrogen production unit can efficiently utilize the excessive power to produce hydrogen fuel that is clean energy and free of carbon emission.

The hydrogen production is a combination of exothermic and endothermic reactions, which operates on low grade, medium grade and high grade heat. Assuming that the regenerative heat cycle utilizes 50% of the heat generated within the unit in the form of low grade heat, the remaining 50% is utilized by the unit in the form of medium grade and high grade heat ranging from 380 to 530◦C. This heat can be provided by the excessive power generated in CANDU SCWR

cycle. Water enters the cycle at 25◦C splits into hydrogen and oxygen with a ratio of 1:8, with 1 part of hydrogen and 8 parts of oxygen leaving the cycle at 25◦C.

5.3 Communication Strategy

CANDU SCWR systems consist of:

- Reactor coolant system
- Feed water system
- Steam system
- Turbine generator system

Flow, temperature, pressure, level are the main variables required to be monitored and controlled, such as e.g. feed water flow, reactor pressure, reactor coolant level during subcritical pressure, core outlet temperature, reactor power etc. A CANDU SCWR communication system would be formulated with the assumption that it operates at base load (not the load following mode). Power and load variations affect the reactor steam generation, e. g. if reactor power demand goes down from 100 per cent to 90 per cent, it would initiate the automated actions of the integrated system i.e. following the control signals of control network, the reactor power control system inserts the control rods in reactor, which results in the reduction of core thermal power, and also the core outlet temperature, steam flow, and reactor pressure. The turbine control valve would also respond according to the decreased load which results in the opening of the relief valve, re-establishing the required nominal steam pressure.

5.3.1 Communication Logic

The communication strategy logic is divided into two broader categories.

- Power network communication logic
- Copper Chloride communication logic

The communication logic builds a safety base for monitoring the CANDU SCWR hydrogen system critical parameters and as the two systems are integrated so any change in the CANDU SCWR loop can be pivotal for system safety and reliability. The communication logic is based on sensors, actuators and controllers communicating via a network to monitor various parameters. Delays in a communication network play an important role for monitoring the system. In the next section, an analysis of the delays of the communication protocols between sensor, actuators and plant control modules is presented.

5.4 System Modeling

In this control logic the standard sensor, controller and actuator model for CANDU SCWR hydrogen co-generation model were used.

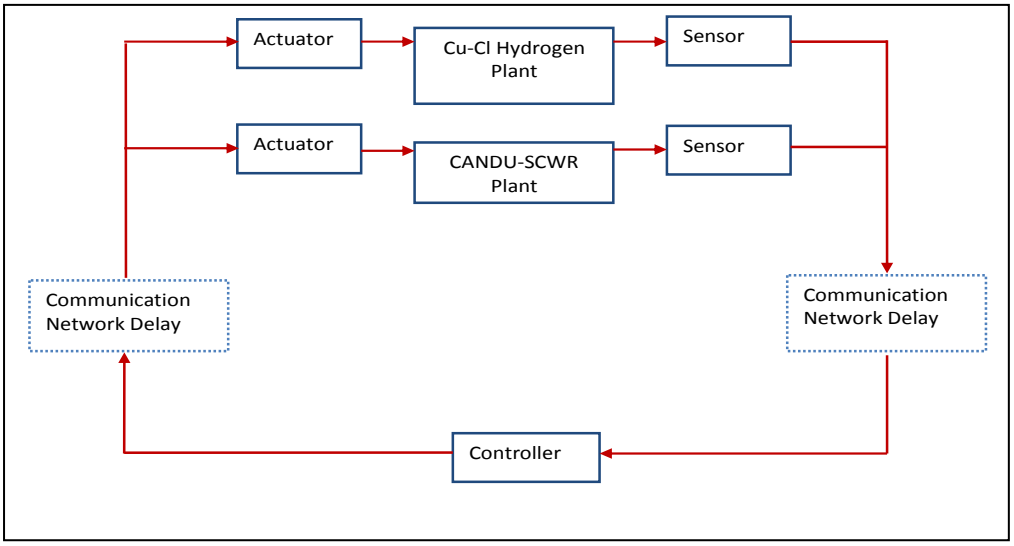


Fig. 5.4.1 CANDU SCWR hydrogen Co-generation

communication model

The similar logic is used for various components of reactor and hydrogen unit. It is assumed that sensor nodes compute the errors between the set points and actual values before forwarding it to the actuator. The controller has set points to compare it with the sensed values and generate control signals. This logic forms a discrete time control system that monitors the dynamic CANDU SCWR hydrogen plant

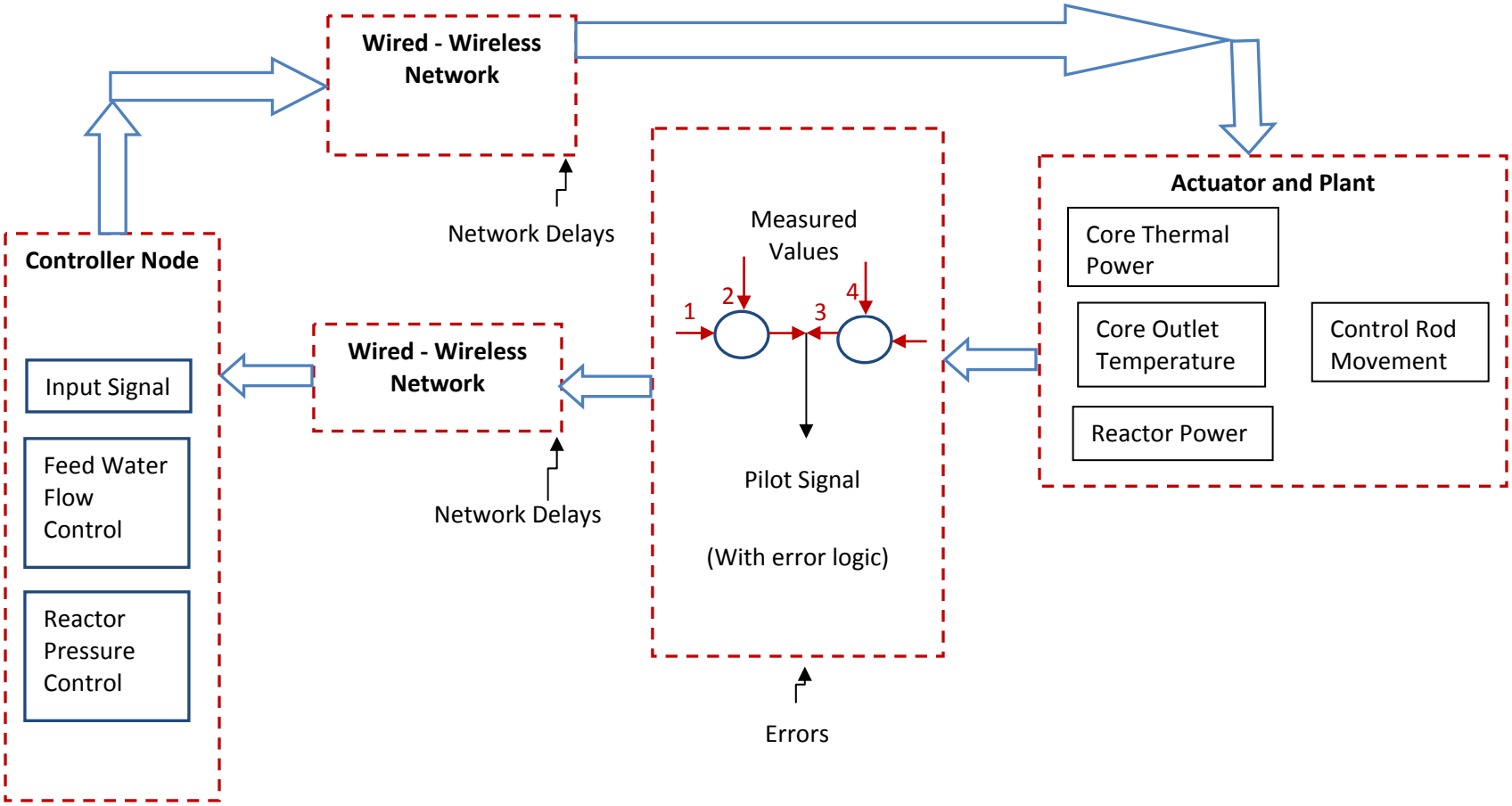
The power communication logic of CANDU SCWR hydrogen plant is shown in Fig. 5.4.1. The controller, sensor, and actuator blocks are connected via communication network. Core outlet temperature measured value is compared with set point (4), and their difference is the core outlet temperature error (3). Similarly the core thermal power measured value is compared with set point of

thermal power (1) and their difference is thermal power error (2). The temperature is regulated by feed water flow mentioned in the controller node. Thus feed water flow variation can regulate the core outlet temperature. Reactor power is dependent on thermal power and core outlet temperature. In case of load variation, the reactor control absorber rods movement control the reactor power and core thermal power. Pressure of reactor steam has direct effect on core thermal power. The thermal power can be controlled by reactor steam pressure control. In the power control logic temperature control was utilized through feed water flow. The core thermal power can be controlled by reactor steam pressure. Feed water flow and steam pressure variation can control core temperature and reactor power. This general logic can be applied in any plant control strategy.

CANDU SCWR's high temperature, high pressure steam and high temperature and low pressure steam is expanded in oxygen unit and fluidized bed unit in hydrogen copper chloride cycle. In Fig. 5.4.2 the fluidized bed coolant temperature measured value is compared with the coolant temperature set point (1) and any difference between the measured value and set point generates an error in coolant temperature (2). Similarly oxygen unit temperature set point (4) is compared with the measured value and generates an error for any difference in both values (3). The main controllers are oxygen temperature controller and steam temperature controller.

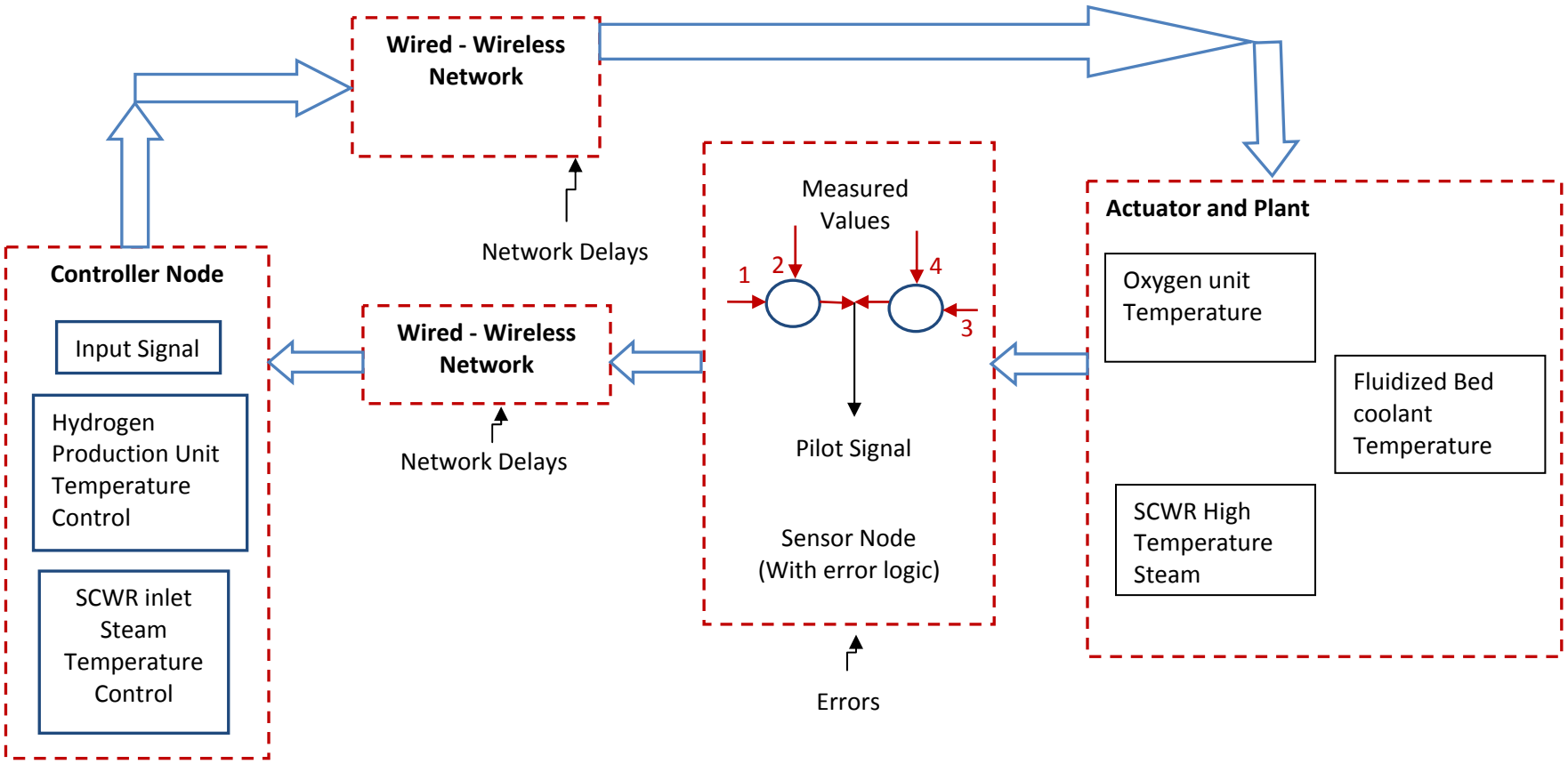
For the communication channel diagrams, a time slot communication channels for specific physical locations was assumed. The first location is for reactor pressure vessel and reactor building. The core outlet temperature can be regulated via feed

water flow and the pressure. The signals are communicated through time slot allocated for error logic control signals which is the difference between the set point values and any variation from specific set point values. A redundant wired and wireless communication path is used to communicate the error signals of the corresponding values to be communicated to the controller node. This co-generation model takes the CANDU SCWR steam to the oxygen unit and the fluidized bed. The combined heat cycle for CANDU SCWR steam for hydrogen production is mentioned in Fig. 5.4.2.



1- Set Point for Thermal Power
2- Thermal Power Error
3- Core Outlet Temperature Error
4- Set Point for Core Outlet Temperature

Fig. 5.4.2 SCWR-Hydrogen Co-Generation power network Communication Channel



1- Set Point Temperature for SCWR steam
2- SCWR Steam Error
3- Oxygen Production Input Steam Temperature Error
4- Set Point for Oxygen Production Input Steam Temperature

Fig. 5.4.3 SCWR-Hydrogen Co-Generation Copper Chloride Communication Channel

5.5 Model Setup

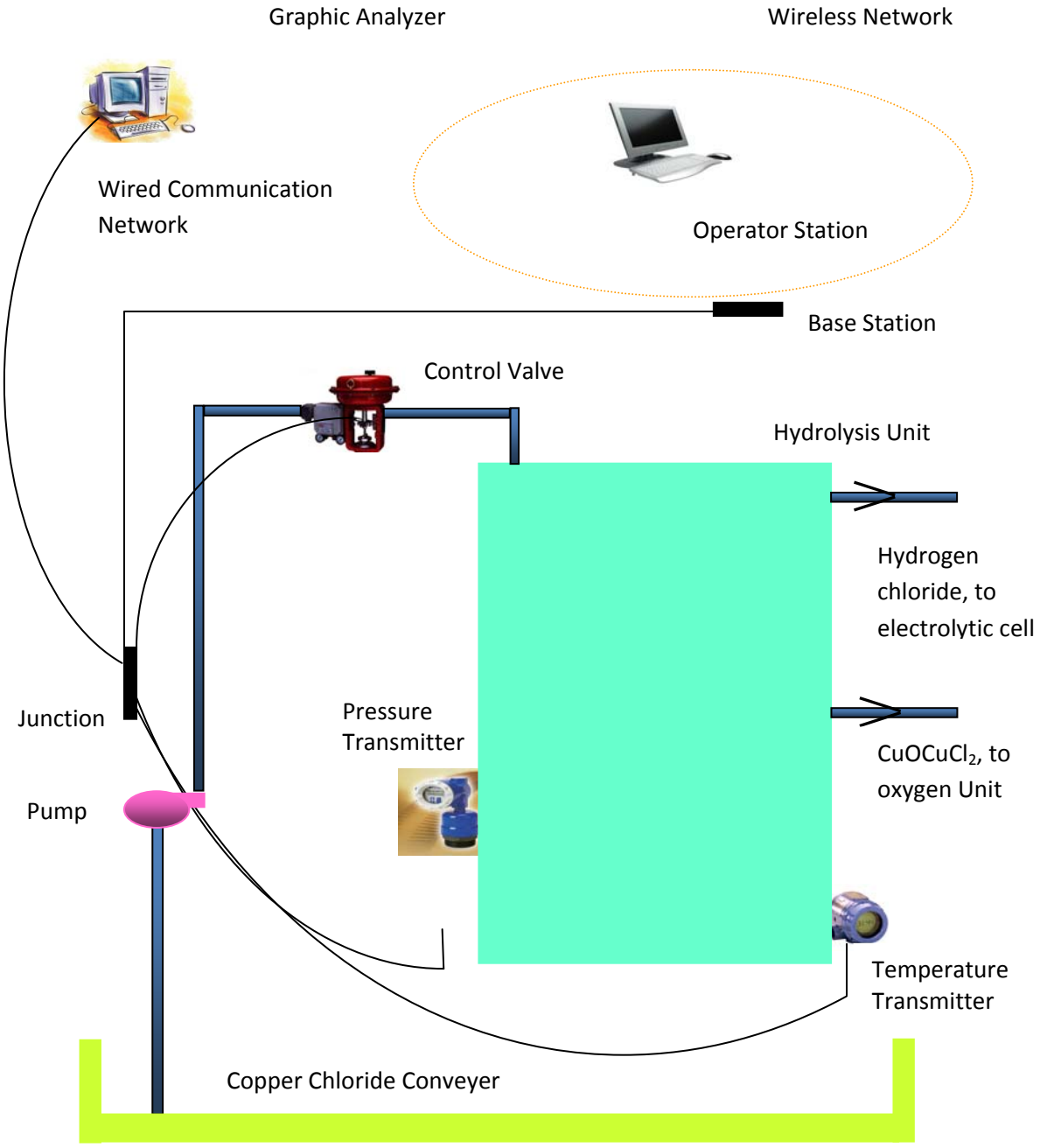


Fig. 5.5.1 Networked Control System of Hydrolysis Unit

The proposed networked communication system contains controller valve, temperature and level transmitter. The sensors placed in the communication model measure temperature and pressure combined together through a junction,

base station transmitter. In the hydrogen research lab at the University of Ontario Institute of Technology (UOIT) mesh networked mote view sensors were used. Installer software can be installed on any laptop or desktop computer. The transmitter, base station, and receiver nodes can communicate via wireless medium, through self healing, multi-hop communication. The sensor and monitor node segregation is shown in Fig. 5.5.2.

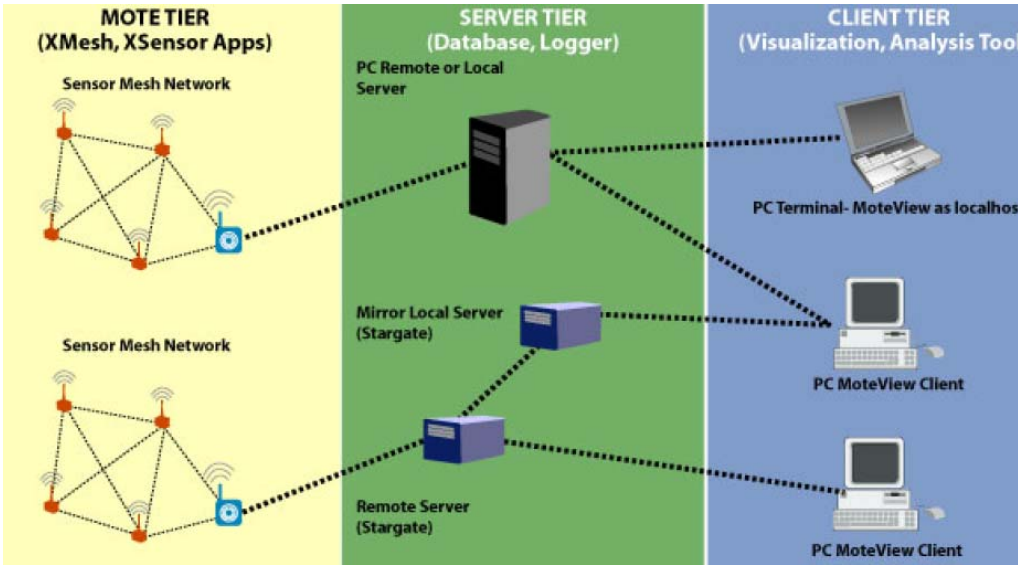


Fig. 5.5.2 Networked Control System of Hydrolysis Unit [42]

Sensors sense the data and direct it to the base station which directs it to the centralized server over a wireless medium. The server configured as Local Area Network (LAN) server and can direct its traffic to the clients within the network. A variation of Zigbee protocol, *XMesh* is used in mote view sensors which is a multi-hop mesh networking protocol that has various options including low-power listening, time synchronization, sleep modes, any-to-base and base-to-any routing.

The delays can be divided into three broader categories:

- Communication Delays
- Computational Delays
- Response Delays (Hardware and Software)

The analog systems used in real time environments have fewer interactions with the software; whereas, the digital control systems are hardware and software interfaced; which works on the standard Open Systems Interconnection (OSI) model. The Networked Control System (NCS) in OSI model works on application, data link and physical layer. The devices operate at the physical layer, the protocol stack is interfaced between the physical and media access layer (MAC), while all the decision support software work on application layer.

The communication delays depend on the communication medium and the specific protocol. The communication model presented in this study only addresses the communication delays as computational and response data were not available. In the future, an evaluation of these delays and a verification of the computations of communication delays by actual measured values over a physical and wireless medium is planned.

The data can be acquired locally (wired) and remotely (wireless). Figure 5.5.1 shows the local data acquisition at graphic analyzer node and remotely at the operator station. For remote data acquisition, the UOIT existing intranet will be used. UOIT has various intranet domains that can be accessed by a secured login

identification and password. The “on campus” domain is widely used by students, faculty and laboratory managers. The temperature, pressure and flow rates can be sensed by local wired sensors and transmitted to the intranet. Industrial vendors are providing wireless sensors with built-in web-enabled protocol interface. These sensors can be used in both wired and wireless communication networks. A special user group for additional storage and user permissions to store and access data can be formed that can access and monitor the parameters anywhere within the campus. Virtual Private Network (VPN) facilities are on UOIT campus. Data can be monitored across the globe through VPN and the internet.

This research’s scope was to analyze the delays in wired and wireless medium that can affect the real time monitor and control system.

6. COMPUTATIONS AND SIMULATIONS

This chapter computed delays involved in the communication model. These delays are computed for protocols, latency, context switching, and queues. In this analysis the researchers used both historical and predictive approaches. The historical computations are compiled through Java and Matlab coding based on the information of protocols frame, communication channel access, and the hardware and software communication based on an OSI model. The predictive analysis is completed using dymonda software which ensures probabilistic risk assessment for a combination of inputs. The predictive analysis shows intermediate failure and complete failure for communication nodes.

IEEE working groups organize and devise working policies for various communication protocols. Distributed Coordination Function (DCF) and Point Co-ordination Function (PCF) are variation of Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA), having RTS and CTS mechanisms that rectify the hidden terminal problems. Chapter three defined the protocols' structure and their operational delays are computed in detail. The protocols are computed according to frame size and format of transmission i.e. two way or four way. The frames' delays in Table 6.1a are computed for Frequency Hop Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) and Distributed Co-ordination Function (DCF). These frames are standard transmission format that lies within IEEE 802.11 a/b/g/e frame format. In all the computations the study took into account the delays involved within the OSI layer particularly at both the MAC and physical layer. The spacing within the protocols contains frames such as the

DCF Inter Frame Space (DIFS), Short IFS (SIFS), and an acknowledgement (ACK) frame. It is worth to note that we are assuming request to send (RTS) and clear to send (CTS) delays in 4-way communication protocol while 2-way communication does not include these delays. The cumulative header formed is 608 micro seconds for two way and 1152 for the four way computations. In all table computations the study applied the bits contained in the frame format, the acknowledgment frame, and propagation delays for calculations. The headings of the table define the table values that are computed for different data sizes with same data rates and with different data rates. For PCF, an additional beacon signal bearing a time delay of 10 micro seconds was used. These calculations are assumed over a distance of 50 metres outdoor communication.

Other researchers [44-50] have generated results that match the calculations mentioned in this research. The calculations for this research assumed an M/M/1 non-contention pre-emptive model, with standard back off mechanism (in case of collision). The innovation of this research is the analysis of all standard protocols for wireless communications with different data rates and different sizes of packets, which according to existing knowledge, is the first effort to analyse wireless protocol behaviour. There exists a non-linear relation between the data rates, data size and delays therefore researcher used log-scale in appropriate simulations to deduce meaningful results.

Table 6.1a IEEE 802.11 Computed Delays

Data Size (Bits)	Two Way FHSS	4 Way FHSS	2 Way DSSS	4 Way DSSS	802.11b DCF 2way	802.11b DCF 4way
246	514.5640	843.1280	418.5460	711.100	418.2490	700.2490
1000	1268.5640	1597.1280	1172.5460	1465.10	1172.2490	1454.2490
1500	1768.5640	2097.1280	1672.5460	1965.10	1672.2490	1954.2490
2000	2268.5640	2579.1280	2172.5460	2465.10	2172.2490	2454.2490
2500	2768.564	3097.1280	2672.5460	2965.10	2672.2490	2954.2490
3000	3268.5640	3597.1280	3172.5460	3465.00	3172.2490	3454.2490
6000	6268.5640	6597.1280	6172.5460	6465.00	6172.2490	6454.2490
10000	10268.5640	10597.128	10172.546	10465.0	10172.2490	10454.2490

Table 6.1b IEEE 802.11 Computed Delays

Data Size (Bits)	802.11b DCF 4way	802.11b PCF	802.11a DCF2Way	802.11a DCF4way	802.11a PCF	802.11g/e DCF2 way
246	700.2490	418.2820	408.2490	712.3970	425.2820	396.2490
1000	1454.2490	1172.2820	1162.2490	1466.397	1179.2820	1150.2490
1500	1954.2490	1672.2820	1662.2490	1966.397	1679.2820	1650.2490
2000	2454.2490	2172.2820	2162.2490	2466.297	2179.2820	2150.2490
2500	2954.2490	2672.2820	2662.2490	2966.397	2679.2820	2650.2490
3000	3454.2490	3172.2820	3162.2490	3466.397	3179.2820	3150.2490
6000	6454.2490	6072.2820	6162.2490	6466.397	6179.2820	6150.2490
10000	10454.2490	10172.2820	10162.2490	10466.397	10179.2820	10150.2490

Table 6.1c IEEE 802.11 Computed Delays

Data Size (Bits)	802.11g/e DCF 4way	802.11g/e PCF
246	688.2500	407.2820
1000	1442.2500	1161.2820
1500	1942.2500	1661.2820
2000	2442.2500	2161.2820
2500	2942.2500	2661.2820
3000	3442.2500	2794.2820
6000	6442.2500	6161.2820
10000	10442.250	10161.2820

The computed delays are in microseconds and are computed for 16 bits, and a 64 bits address with different data size and different data rates.

Table 6.2a IEEE 802.15.4 Computed Delays

Data Size in Bits	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay
8.0000	20.0000	8.6200	11.0200	40.0000	2.8730	3.6730
16.0000	20.0000	11.8200	14.2200	40.0000	3.9400	4.7400
80.0000	20.0000	37.4200	66.3660	40.0000	12.4730	13.2700
100.0000	20.0000	45.4200	47.8200	40.0000	15.1400	15.9400
120.0000	20.0000	53.4200	55.8200	40.0000	17.8100	18.6000

Table 6.2b IEEE 802.15.4 Computed Delays

Data Size in Bits	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay
8.0000	60.0000	2.8730	3.6730	80.0000	2.1550	2.7550
16.0000	60.0000	3.9400	4.7400	80.0000	2.9550	3.5550
80.0000	60.0000	12.4730	13.2700	80.0000	9.3550	9.9550
100.0000	60.0000	15.1400	15.9400	80.0000	11.3550	11.9550
120.0000	60.0000	17.8100	18.6000	80.0000	13.3550	13.9850

Table 6.2c IEEE 802.15.4 Computed Delays

Data Size in Bits	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay
8.0000	100.0000	1.7240	2.2040	120.0000	1.4366	1.8370
16.0000	100.0000	2.3640	2.8440	120.0000	1.9700	2.3700
80.0000	100.0000	7.4840	7.9640	120.0000	6.2370	6.6367
100.0000	100.0000	9.0840	9.5640	120.0000	7.5700	7.9700
120.0000	100.0000	10.6840	11.1640	120.0000	8.9030	9.3030

Table 6.2d IEEE 802.15.4 Computed Delays

Data Size in Bits	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr Delay	Data Rate	Zigbee 16 addr Delay	Zigbee 64 addr delay
8.0000	140.0000	1.2314	1.5743	160.0000	1.0775	1.3775

16.0000	140.0000	1.6886	2.0310	160.0000	1.4775	1.7775
80.0000	140.0000	5.3460	5.6886	160.0000	4.6775	4.9775
100.0000	140.0000	6.4886	6.8314	160.0000	5.6775	5.9775
120.0000	140.0000	7.6314	7.9743	160.0000	6.6775	6.9775

Table 6.3 IEEE 802.11 Computed Delays

Data Size in Bits	Ack	Phy	Mac Header	Mac footer	Twait
8.0000	4.4000	1.2314	1.2314	1.5743	1.0775
16.0000	2.2000	Tsifs	Tslot	TLifs	Tsifs
80.0000	1.4600	1.3775	280 bits	160 bits	1.3775
100.000	1.1000				
120.000	0.8800				

Table 6.4 Computed Software Delays [19]

$T_{RTL\ (Idle)Max}$	$T_{RTL\ (Idle)Max}$	$T_{RTL\ (Idle)Avg}$	$T_{RTL\ (Idle)Avg}$	$T_{RTL\ (Idle)Avg}$
13.5 (Latency)	33.1 (Context Switch)	1.7 (Latency)	8.7 (Context Switch)	8.7 (Context Switch)
$T_{RTL\ (Busy)Max}$	$T_{RTL\ (Busy)Max}$	$T_{RTL\ (Busy)Avg}$	$T_{RTL\ (Busy)Avg}$	
196.8 (Latency)	193.9 (Context Switch)	5.4 (Latency)	15.4 (Context Switch)	

The formulae in Section 3.4, Chapter 3, and Tables in section 3.5 are used to compute delays for various protocols. The study plotted the delay computations against the data to show the effects of increasing data size and data rates.

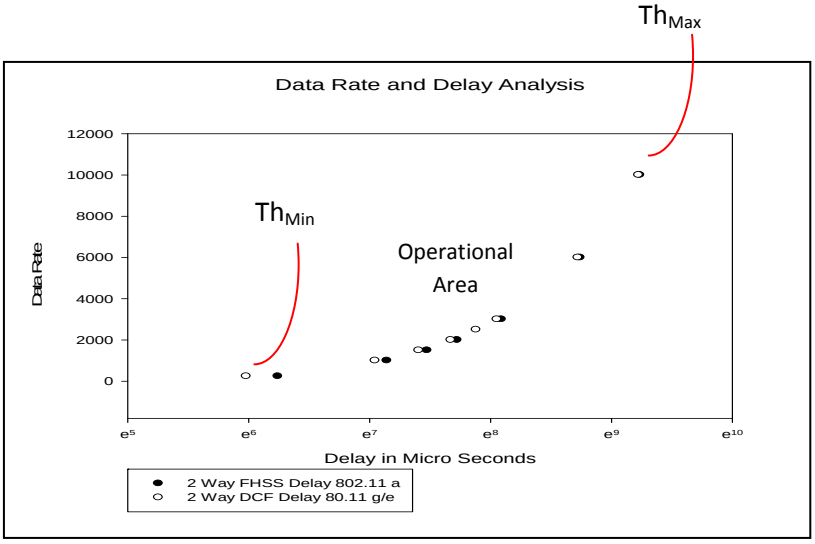


Fig. 6.1 Data Rate and IEEE 802.11 corresponding Delay in Micro-seconds

The simulation uses "log base 10 values" on the x-scale to explain the relation between increasing data rates with increase in data packet size, corresponding to increase in delay. The protocol has minimum and maximum transmission capacity defined as Th_{min} (minimum packet size) and Th_{max} (maximum packet size) and with increasing data packet size the difference between protocols decreases and the increase in delay indicates the exhaust capacity of the IEEE 802.11 protocol.

Fig. 6.1 shows that the increase in data rate diminishes the delay difference between various IEEE 802.11 protocols. Any of 802.11 wireless protocols for a higher data rate can be selected. The results were generated using different data size iterations. As an increase in data size iterations, the delay difference between protocols decreases. The results show increase in delays despite an increase in

data rate, and the reason behind the simulation result is that every protocol frame has data size limitation and despite an increase in data rate, the delay grows exponentially due to fixed frame size and reaches an infinite value. The horizontal log scale was chosen to keep the variation smoother as the results generated through normalized scales for data rate were not conclusive. It was decided to use the log scale on the horizontal axis to explain the results to the reader. The terms Threshold minimum and Threshold maximum are used for the delays in protocols for the predictive model. These values show the limits for minimum size of the data for a particular protocol, between these values is the operational area of the protocol. It is important to mention that log values act as filters to eradicate the variation in the corresponding values.

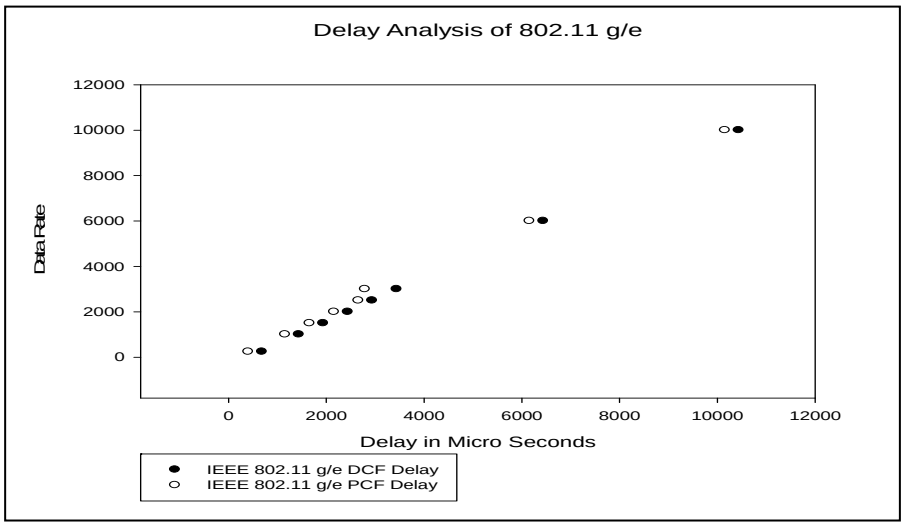


Fig. 6.2 Data Rate and IEEE 802.11g/e corresponding Delay in Micro-seconds

The result in Fig. 6.2 shows linear propagation without log-scale and there is no indication of the protocol threshold points in the simulation as it was the case in

Fig. 6.1. The Fig. shows that PCF delays are lesser than DCF. PCF is contention free access mechanism in 802.11 g/e protocol that is well suited for real time control and monitor activities. In this scenario, the same situation applies as described previously for Fig. 6.1. The only difference between the two is that normal scale in x-axis for delay was chosen. In this scenario the delays due to header, trailer, hand shaking and acknowledgement are not an issue as both PCF and DCF have similar frame format; therefore, only the smaller data difference between the two frame formats and only the need to show that for real time control network, PCF is better than DCF if operated at higher data rates were chosen. Similarly in other cases the researcher adapted the same strategy to use log scaling and normal scaling on x or y axes as is suitable to explain the results to the reader. It is not anticipated that the reader has enough knowledge specific background to understand why a different scaling was adapted for each simulation.

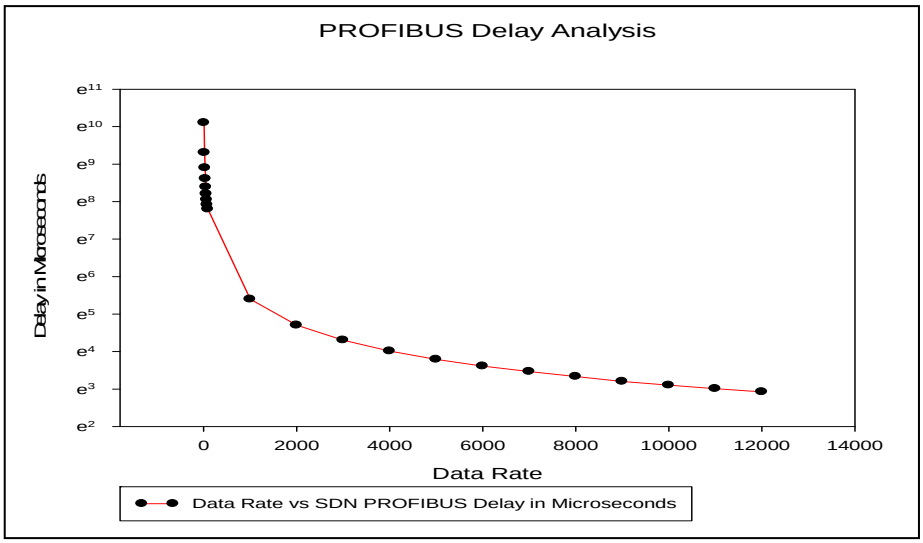


Fig. 6.3 PROFIBUS Data Rate and Corresponding Delay in Micro-Seconds

Fig. 6.3 shows that the increase in data rate minimises the delay. As previously observed the higher data rates can lower the delays.

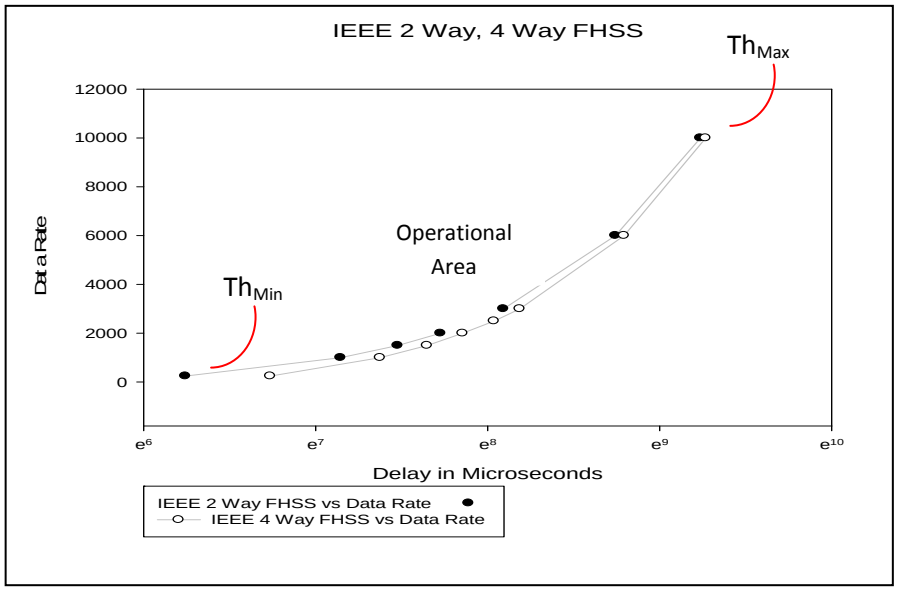


Fig 6.4 IEEE 802.11 FHSS Delay in Micro-seconds

In this simulation, the same "log base 10" x-scale is used with Th_{min}, Th_{max}, and the operational area of the protocol is mentioned. The results are similar to the Fig. 6.1 simulation.

Fig. 6.4 shows that 4 way FHSS system with ACK, RTS and CTS. Four way handshaking transmissions are always a system overhead that can be overcome by higher data rates.

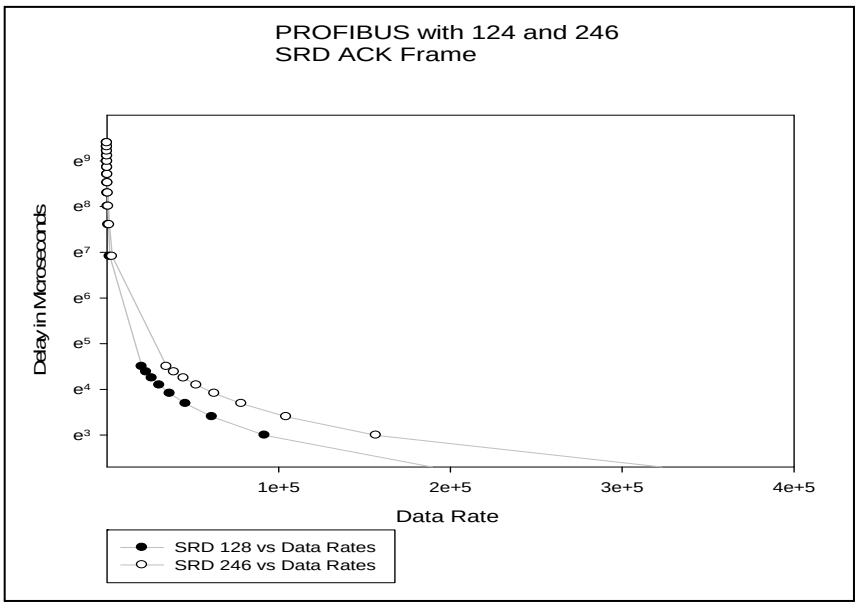


Fig. 6.5 PROFIBUS Delay in Micro-seconds

Fig. 6.5 shows a comparison of 128 bits and 246 bits ACK frame PROFIBUS transmission. Larger acknowledgement bits can enhance delays.

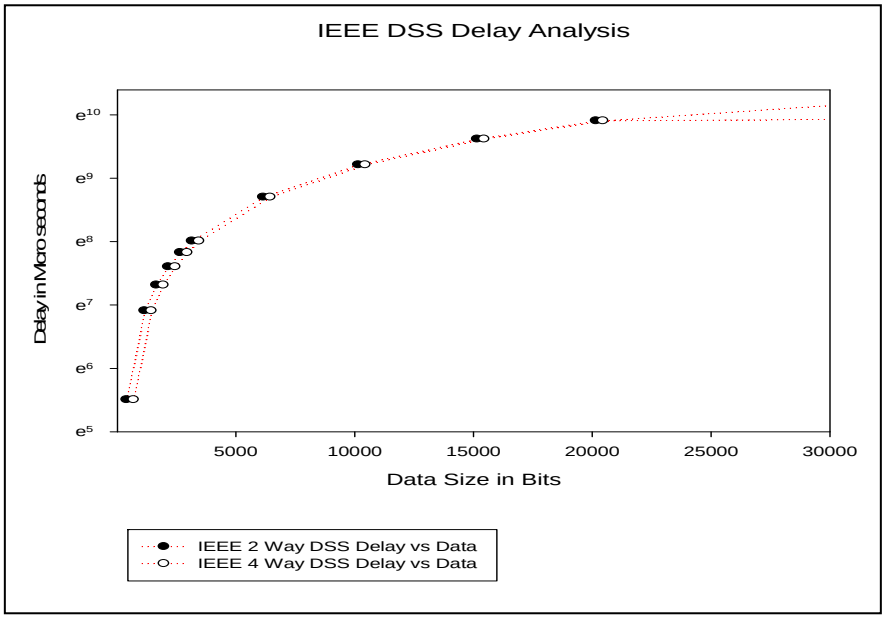


Fig. 6.6 IEEE 802.11 4 way and 2 way Delay in Micro-Seconds

It was observed in all simulations that 4-way transmission with acknowledgement provides data transmission reliability, but it is a system over head due to larger delays. Future discussions should analyze two way retransmission algorithms that can minimize the system delay.

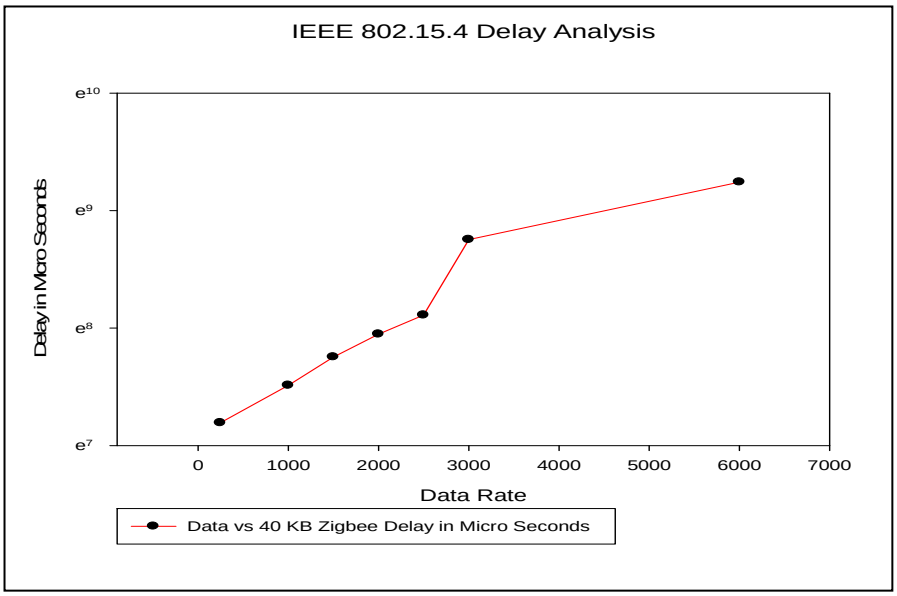


Fig. 6.7 IEEE 802.15 4 Delay in Micro-seconds

Fig. 6.7 is simulation of 802.15.4 protocol for fix size data transmission with different data rates. The simulation shows that for a fixed data size the delays decrease with higher transmission rates, and it is also shown that data size has lesser impact on delays if transmission rates are high.

6.1 CANDU SCWR Hydrogen DFM Model

The DFM model design of the CANDU SCWR hydrogen co-generation loop, analysed the main components of the loop. This analysis parameters are flow, level, and power. The main architecture consists of sensors, controllers, and actuators. The study also defined various conditions for pumps, valves, level controllers and actuators that generate a set of prime implicants.

As previously defined in fault tree analysis, the major failure modes are:

- Improper condensate supply to LP heaters

- Improper condensate supply to De-aerator
- Improper steam supply to HP heaters
- Super heated steam loop failure
- Loop1 high temperature, high pressure failure
- Loop2 high temperature, low pressure failure
- Oxygen reactor unit failure
- Hydrogen unit failure

Fig. 6.1.1 shows the snapshot of the model, showing the components of the model. The prime implicants generated through the deductive analysis for BFW pump failure; hydrogen reactor failure and SCWR loop high temperature failure.

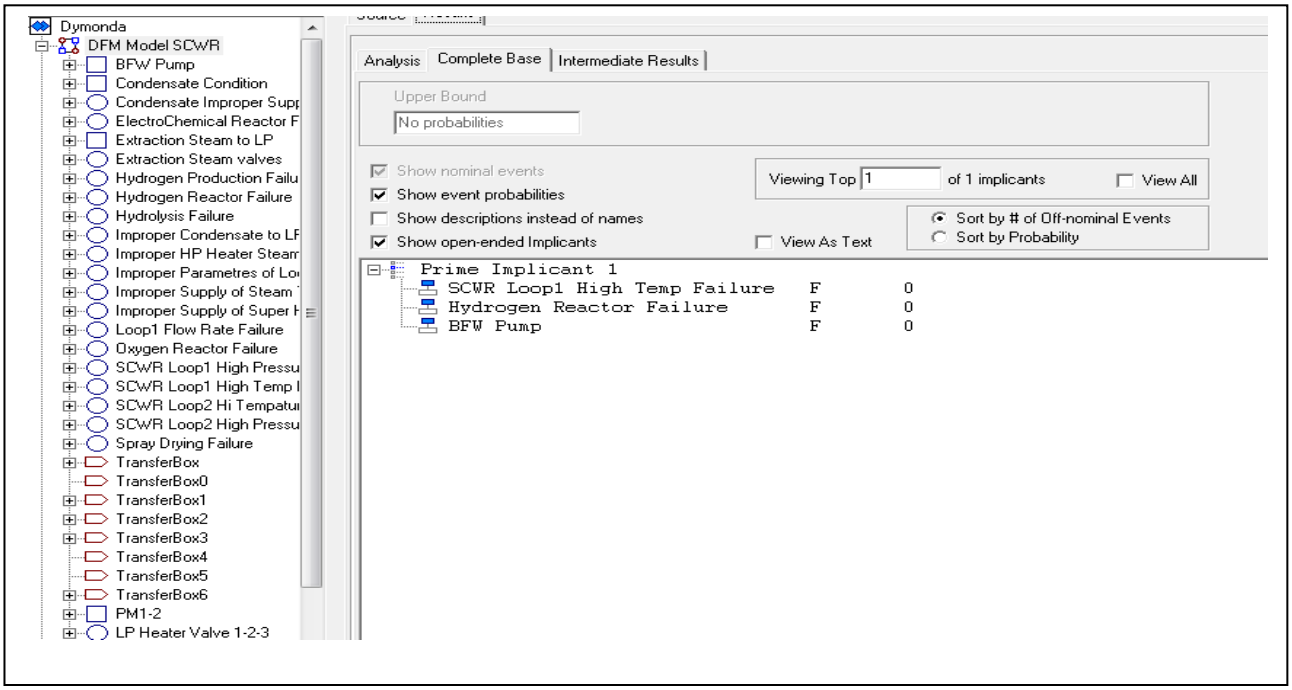


Fig. 6.1.1 CANDU SCWR Hydrogen Co Generation Model

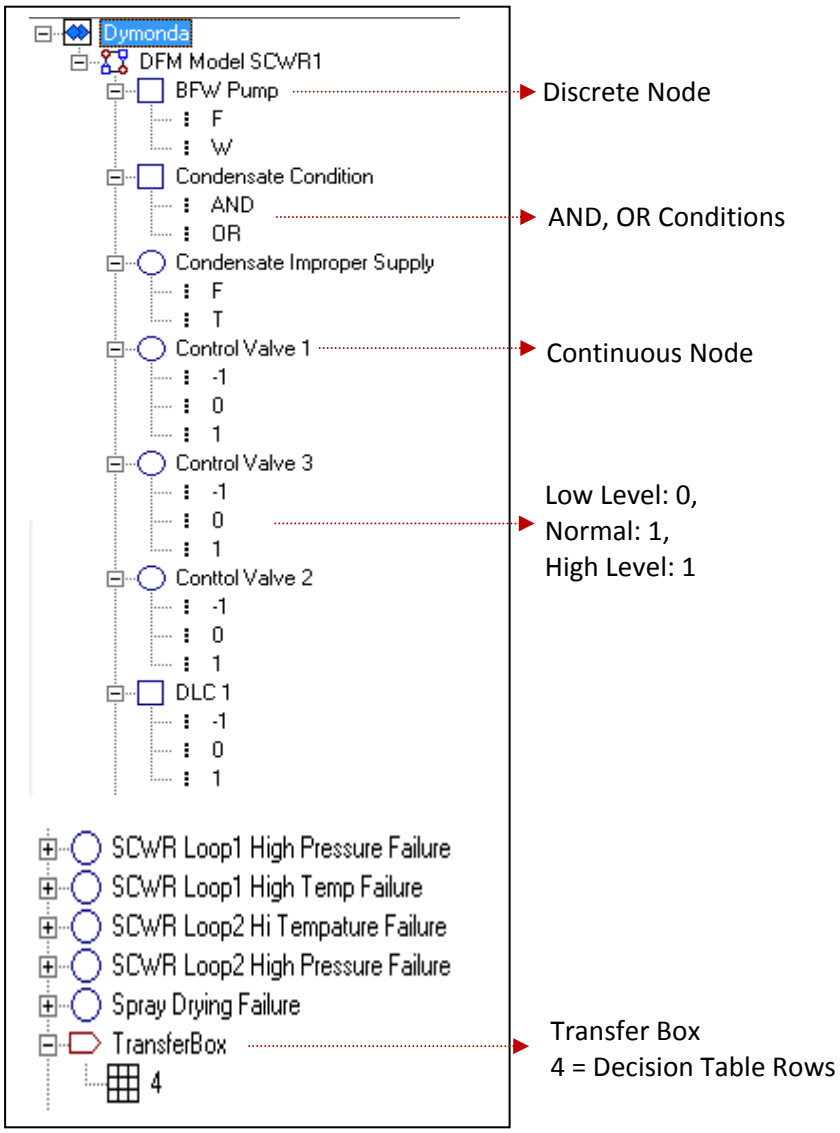


Fig. 6.1.2 Components of CANDU-SCWR Hydrogen Co-Generation Model

Fig. 6.1.2 shows the discrete node of BFW pump that has two, failed and working conditions associated with the node. Condensate condition discrete node with AND, OR switch conditions. Continuous node of control valve 1 with low, high and normal conditions. Transfer box with multiple combination of four different condition rows in a decision table.

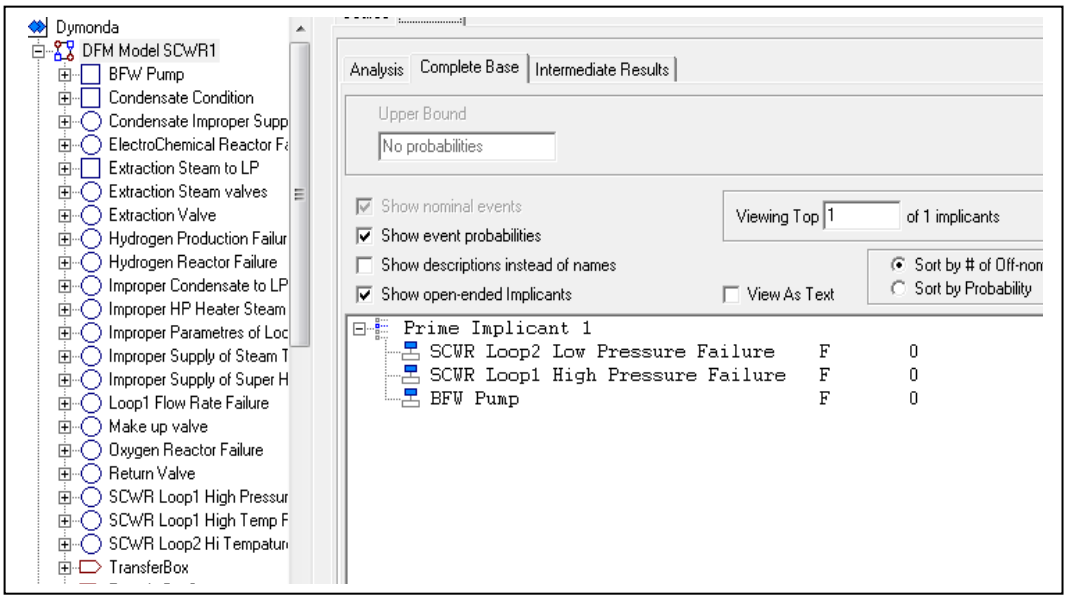


Fig. 6.1.3 Components of CANDU-SCWR Hydrogen Co-Generation Model

Fig 6.1.3 shows the prime implicants generated through deductive analysis. SCWR loop1 and loop2 and BFW pump.

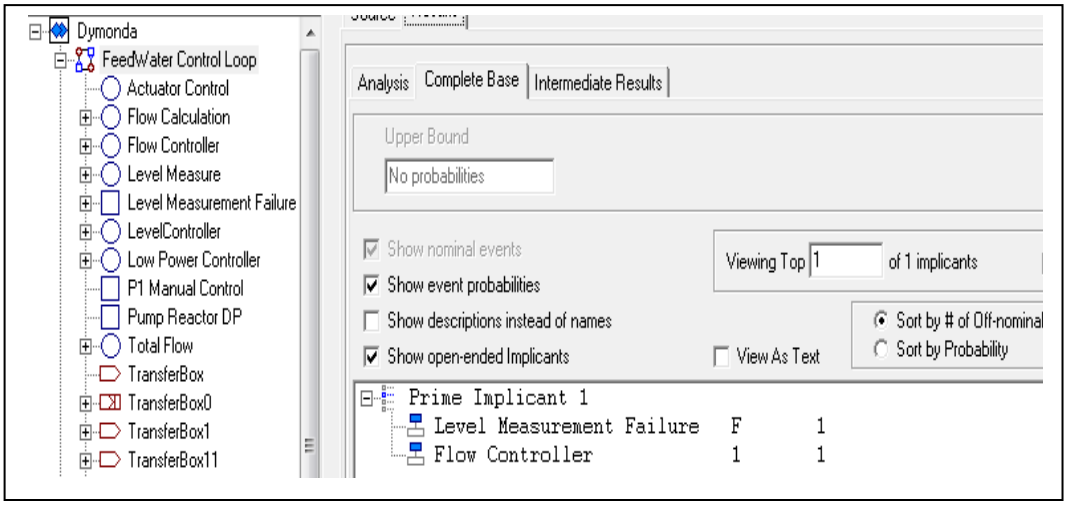


Fig. 6.1.4 Feedwater Controller Actuator and Sensor loop

Fig. 6.1.4 shows the CANDU-SCWR hydrogen co-generation feed water control loop that is the main source of heat for the hydrogen production unit. The prime implicants generated through deductive analysis are Level measurement failure and flow controller.

In the dymonda DFM model, the communication system delays were modelled. Application layer is the upper most layer on OSI model, and permissible and exception level of delays on the OSI application layer are defined. In concurrent programming, various applications are working in parallel and are assigned thread numbers from 1 to n. These threads can be prioritized or non-prioritized, synchronized or asynchronized. Every thread is timed in and timed out by thread server application which throws an exception on thread time out and takes in another application in a round robin mechanism. Polling is done through an echo server to all of the client applications. Parallelization can also be achieved at physical layer through multiple processors, by adding new hardware interfaced

through Message Passing Interface (MPI). Parallelization at hardware level is expensive and requires additional interfaces and programming skills to compute. Recent advancements in Very Large Scale Integration (VLSI) micro chip technologies have introduced Personal Computers (PC) that can perform exponential computations. Flynn's taxonomy of parallel programming is a classification method for instructions and data processing. The taxonomy classifies four distinct features for instructions and data. Flynn's taxonomy was initially developed for hardware interfaces but it is recently adapted for single processor multi-data and instructions architecture. Various GUI tools like java and dot net technologies have built in libraries for threads programming. In this model it is assumed that the software parallelized model can run multiple context switched programs.

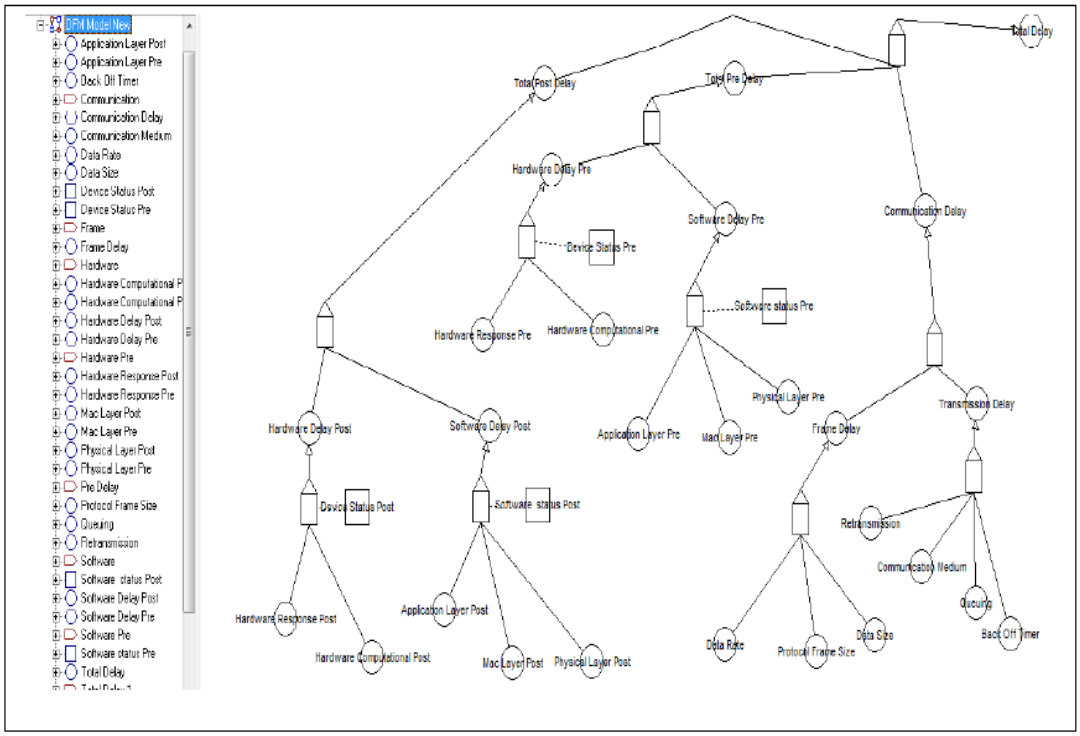


Fig 6.1.5 Communication Network DFM Model

For software delay defined conditions are threshold minimum, threshold maximum and normal are defined. The communication between software and hardware takes place through media access layer. The Mac layer delay is commonly associated with the communication between the concurrent programs and protocols. The Mac layer levels defined are permissible and exception. Real time control systems must respond within specific time limit and any high order delay can throw an exception for a timed out thread. Software and hardware interface for controller, sensor or actuator can be interfaced through network interface card. The command signals coming in from the sensor, controller or actuator and the signals going out from the software are time in and time out entities respectively. Software delays are associated with application, Mac and physical layer.

Fig. 6.1.5 shows the DFM model of the system delays. The study analyzed the various states of the DFM model nodes and derived an inductive and deductive analysis using the combination of states of all types of delays. For hardware delay, the conditions as threshold minimum, threshold maximum and normal were defined. Hardware delay is a combination of response delay, and computational delay. The response time is associated with the host (main BIOS controller), slave (dependent devices) and bridge (hardwired interfaces) Interrupt Requests (IRQ) for services, the modelled response time delays are: in time, and exception. Computational delay depends on the job duration, and the CPU clock cycle. As computational delays are associated with variable source; they are categorized as

permissible and not permissible. The total hardware delay is defined as threshold maximum, minimum and normal.

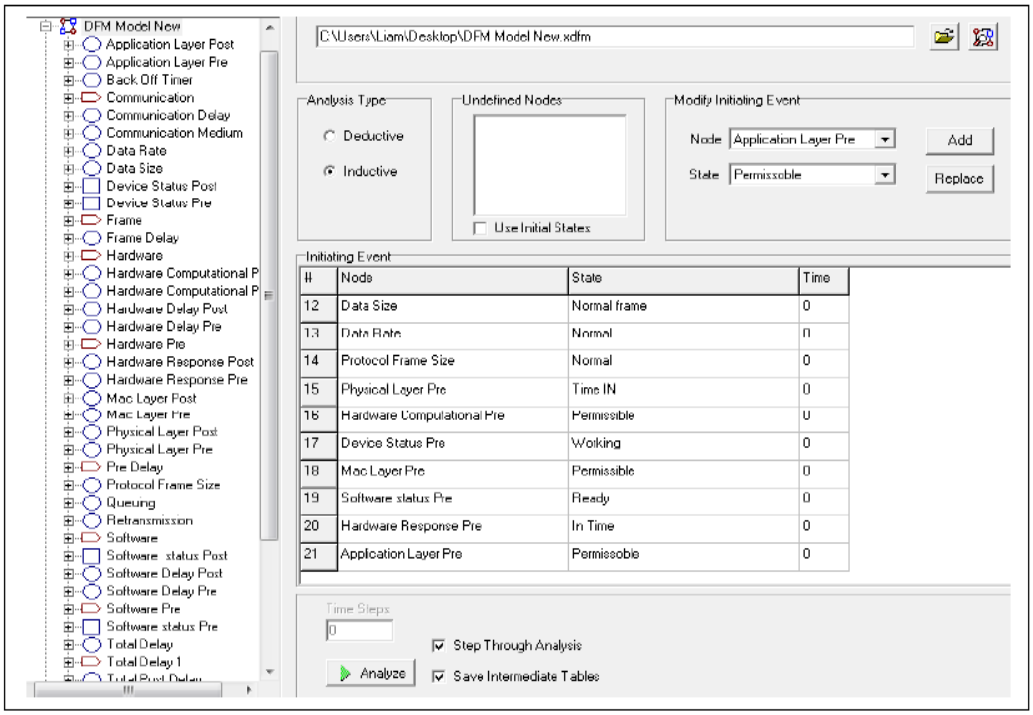


Fig 6.1.6 Communication Network DFM Inductive Analysis

Fig. 6.1.6 shows the inductive analysis of the DFM network delay model. The analyzed model with multiple conditions can generate more than hundred combinations. The discrete values produce top events in the form of prime implicants that are critical for a the network system. The prime implicants generated through the dymonda software in Fig. 6.1.7 are retransmission, queuing, protocol frame size, data size, data rate and communication medium.

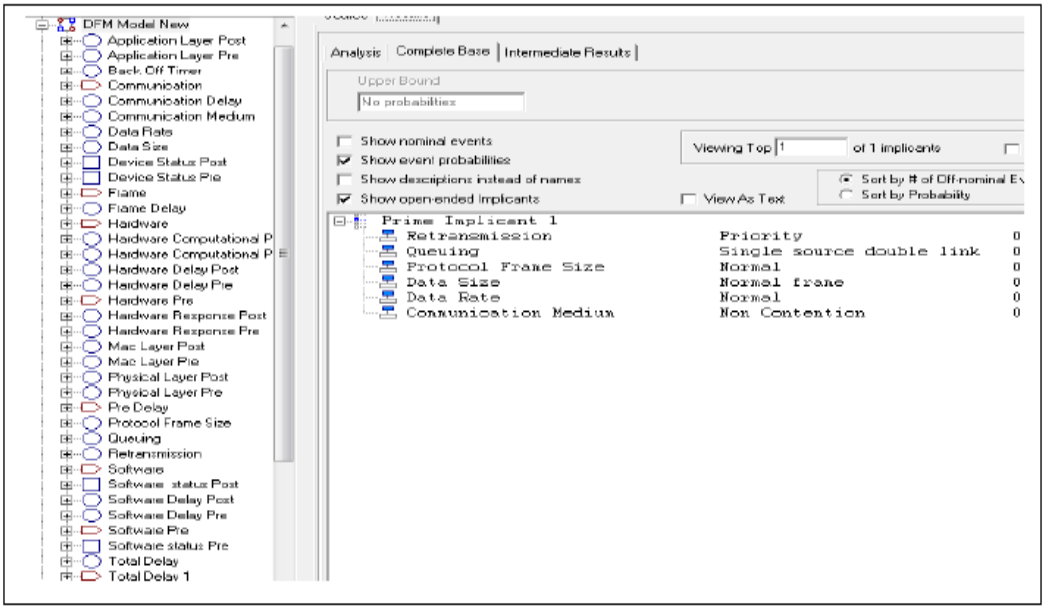


Fig. 6.1.7 Communication Network DFM Inductive Prime Implicants

The conditions selected were:

Priority for retransmission.

Single source double link for queuing.

Normal size for protocol frame.

Normal size for data frame.

Normal rate for data rate.

Non contention for communication medium.

For frame delay the conditions were set as permissible or exception. For Frame delay the conditions for data rate as normal, medium and high were formed. These rates are dependent on the protocol and the network used. Similarly for protocol frame size, the conditions as minimum, maximum, and normal frame size were adapted. The maximum frame size would have higher delays as

compared to minimum and normal frame size. The size of the data in the frame can also affect delays and were conditioned it as small, normal and medium sized. For every transmission, there was a successful transmission or connection timed out condition. For transmission medium delay, formed the conditions for retransmission as priority and non priority were present. Delays can be higher for non prioritized packets. For communication medium, an adapted contention and non contention strategy was used. Communication medium with contention can have higher delays. The Queuing delays depend on the arrival rate of packet and the service rate of the network. The conditions set for this queuing dymonda model are single source single link double source single link and single source double link. The delays are different for different delaying architecture. In case of a collision the transmission delays depend on back off timer. These delays are higher for multiple collisions. The collision, no collision conditions for back off timer were adapted.

For Frame delay the conditions as permissible or exception were set. For Frame delay, the study formed the conditions for data rate as normal, medium, and high. These rates are dependent on the protocol and the network used. Similarly for protocol frame size the study adapted the conditions as minimum, maximum, and normal frame size. The maximum frame size would have higher delays as compared to minimum and normal frame size. The size of the data in the frame can also affect delays and was conditioned that it was small, normal and medium sized. For every transmission, a successful transmission or connection timed out was realized. For transmission medium delay it was formed that the conditions for

retransmission as priority and non priority. Delays can be higher for non prioritized packets. For communication medium, the study adapted the contention and non contention strategy. Communication medium with contention can have higher delays. The queuing delays depend on the arrival rate of packet and the service rate of the network. The conditions set for the queuing dymonda model are single source single link double source single link and single source double link. The delays are different for different delaying architecture. In case of a collision the transmission delays depend on back off timer. These delays are higher for multiple collisions. It was adapted for the collision, no collision conditions for back off timer.

6.2 Latency Delay

The latency (hardware) delays are associated with the personal computer components used for processing the requests for communication between hardware components through the peripheral component interconnect (personal computer bus) or PCI bus [51].

The main components are the host (Main Controller BIOS), slave (Dependent Devices), and the bridge (Hardwired Interface between IRQs). The interrupt requests are predetermined set of requests with pre-assigned numbers reserved for specific devices. IRQ is a tri-state interrupt mechanism with sample, recovery and turn around phase. IRQ requests are dependent on the CPU clock cycle; the CPU clock cycle delay for one clock cycle is assumed as 170 nano seconds for our system. Vendors have specified their system clock cycles, and we are assuming

the PCI Bus delays for a 25 MHz CPU clock cycle as 3.84 micro seconds and for 33 MHz CPU clock cycle as 2.88 micro seconds [51].

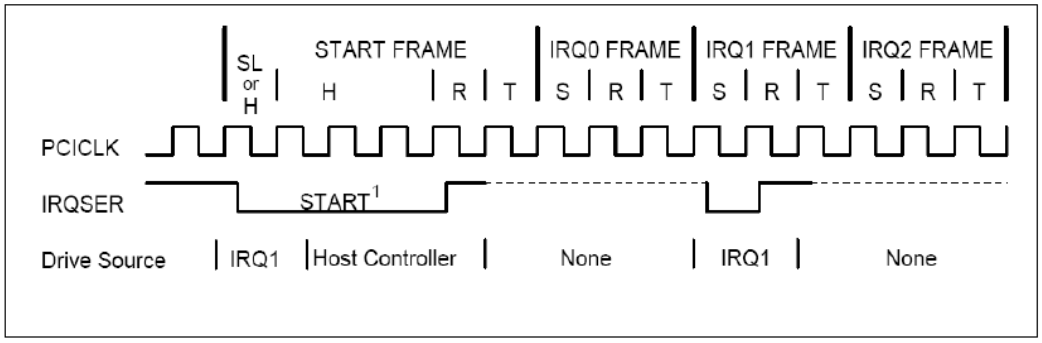


Fig. 6.2.1 shows the IRQ setup for personal computer

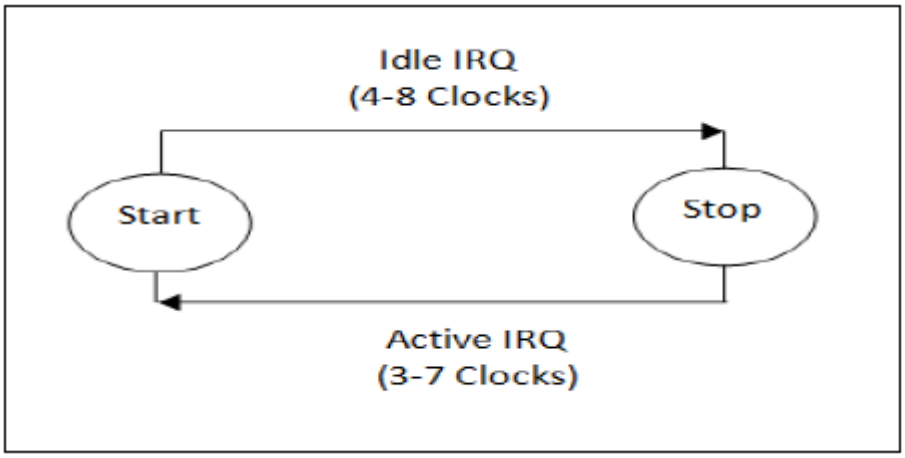


Fig. 6.2.2 shows the IRQ delays between start and stop states

Multitasking as process or thread-based can be distinguished. Various application programs running in parallel at coarse-grain level is process-based multitasking. Thread based multitasking is fine-grained multitasking that involves parts of the programs to run as threads within the program. These threads can run in parallel through independent paths defined by the main program. As previously described, the Flynn's taxonomy has classified parallel programming into four distinct classifications. Single program single data (SPSD), Single program multiple data

(SPMD), multiple program single data (MPSD) and Multiple program multiple data (MPMD). The taxonomy was developed for hardware architecture but by using specialized thread programming tools, advantages of multi-programming can be achieved within an application or a program. The context switching delay can be due to switching between the applications or between threads of the same program. The study programmed an echo server that acts as Master Controller (Server) for all other slave (client) programs. The socket programming in java establishes a bidirectional communication between a server program and one or more client programs. A virtual hardware port number is assigned by the server program that can be used by the clients to connect to a networked program. For multiple clients' connection to server, both server and client should be programmed as multithreaded applications. The server broadcasts its port to the clients for connection and clients echo back their identification to the server for connection.

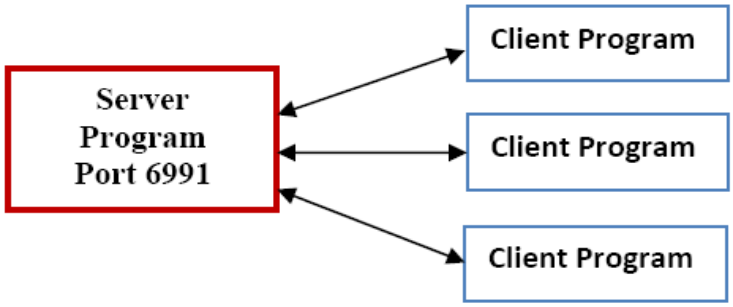


Fig. 6.2.3 Server/Client Echo Connection

Fig. 6.2.3 shows the multi-communication client server environment for echo-server connection. The application for echoing server ports to the clients was programmed. The port we assumed for broadcast is 6991, and a port filter for port

number 6881 (assumed value) was created. For this study both the client and server application as multi-thread client and server echo application were designed, so multiple client programs could run concurrently using port 6991.

The advantages of thread based communication between server and client is that threads share the same memory address space within the program, context switching as communication between threads is inexpensive and CPU cycles are efficiently utilized by the thread programming. Fig. 6.2.4 shows the coding in java on JCreator LE tool [52]. The results show that server echoes the port number ready for client connection. The client echoes its identification to the server and a multi-threaded connection is established through the server virtual port and client virtual IP address.

In a virtual environment, there can be multiple client server architectures on a single machine that enhances the virtual programming concurrent model. The advantage of virtual environment is that one can isolate various clusters as parent and child applications and calculate the delays between the connections. In virtual programming models one parent partition, and several applications that are child partitions may be present. A program that makes a request for services is called client connection and a program that offers requested services from one or more clients is called server connection. The software access delay is governed by two parameters, the CPU cycles and the Input / Output (I/O) operations. CPU cycle involves hardware computing power while I/O operation is subsequently reduced through socket programming. The socket communication is actually server client port bidirectional communication that enables data transmission between server

and client. These ports can be hardware or virtual ports. For this communication delay analysis the context switching can be due to control switching between two client programs controlled by single cluster server or within a program that has built in parent and child thread. The Fig. 6.2.4 shows the virtual socket program environment that resides various applications, in the user mode while the communication bus works in kernel mode between several applications. This application was developed in Jcreator and designed on one server that echoes port number for server connection, and the client that accepts the echo and connects to the server. The application was developed in a VMWare environment that shows the server and client delays within a single machine, on top of a Windows 7 operating system. The delays were different for several runs with a tolerance of ± 3 microseconds.

Fig. 6.2.5 shows the output of server and client echo-server applications, The Server was run in Windows 7 and the client was run in VM-ware player with a Linux (Ubuntu) platform.

```
import java.io.*;
import java.net.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
public class EchoServer extends Thread {
static final String APP_NAME = "EchoServer";
static final int PORT = 6991;
static ServerSocket serverSocket;
static int port = PORT;
Socket clientSocket;
BufferedReader is = null;
BufferedWriter os = null;
public EchoServer(Socket cs) {
clientSocket = cs;
}
public static void main(String args[]) {
if (usageOnly(args)) {
System.exit(0);
}
```

```
initialize(args);
printMsg("EchoServer running on port: " + port + "...Ready to accept connections...");
while (true) {
try {
Socket clientSocket = serverSocket.accept();
EchoServer es = new EchoServer(clientSocket);
es.start();
} catch (IOException e) {
printMsg("Cannot accept client connection.");} } }
public void run() {
processClientRequest();}
private static boolean usageOnly(String args[]) {
if (args.length > 1 || (args.length == 1
&& (args[0].equalsIgnoreCase("-usage")
|| args[0].equalsIgnoreCase("-help")
|| args[0].equalsIgnoreCase("-h")))) {
System.out.println("Usage: java " + APP_NAME + " [<port>]");
System.out.println(" The default port is " + port + ".");
return true;} else { return false;} }
private static void initialize(String args[]) {
processCommandLine(args);
try {serverSocket = new ServerSocket(port); } catch (IOException e) {
printMsg("Cannot create server socket " + "on port: " + port + ". Exiting...");

System.exit(0);}}
private void processClientRequest() {
try {os = new BufferedWriter(
new OutputStreamWriter(clientSocket.getOutputStream()));
is = new BufferedReader(
new InputStreamReader(clientSocket.getInputStream()));} catch (IOException e) {
printMsg("Cannot handle client connection.");
cleanup();
return;
}
try {
String input = is.readLine();
if (input != null) {
input = APP_NAME + ": " + input + " " + getDateTime();
os.write(input, 0, input.length());
os.flush();
}
} catch (IOException e) {
printMsg("I/O error while processing client's print file.");
}
cleanup();
}
private void cleanup() {
try {
if (is != null) {
is.close();
}
if (os != null) {
os.close();
}
if (clientSocket != null) {
clientSocket.close();
```

```
}
} catch (IOException e) {
printMsg("I/O error while closing connections.");
}
}
private static void processCommandLine(String args[]) {
if (args.length != 1) {
return;}
port = Integer.parseInt(args[0]);
if (port < 1 || port > 6881) {
port = PORT;
printMsg("Using port " + port + " instead.");
}}
private static void printMsg(String msg) {
System.out.println(APP_NAME + ": " + msg);}
private String getDateTime() {DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd
HH:mm:ss");
Date date = new Date();
return dateFormat.format(date);}}
```

Fig. 6.2.4 Java Echo-Server Application

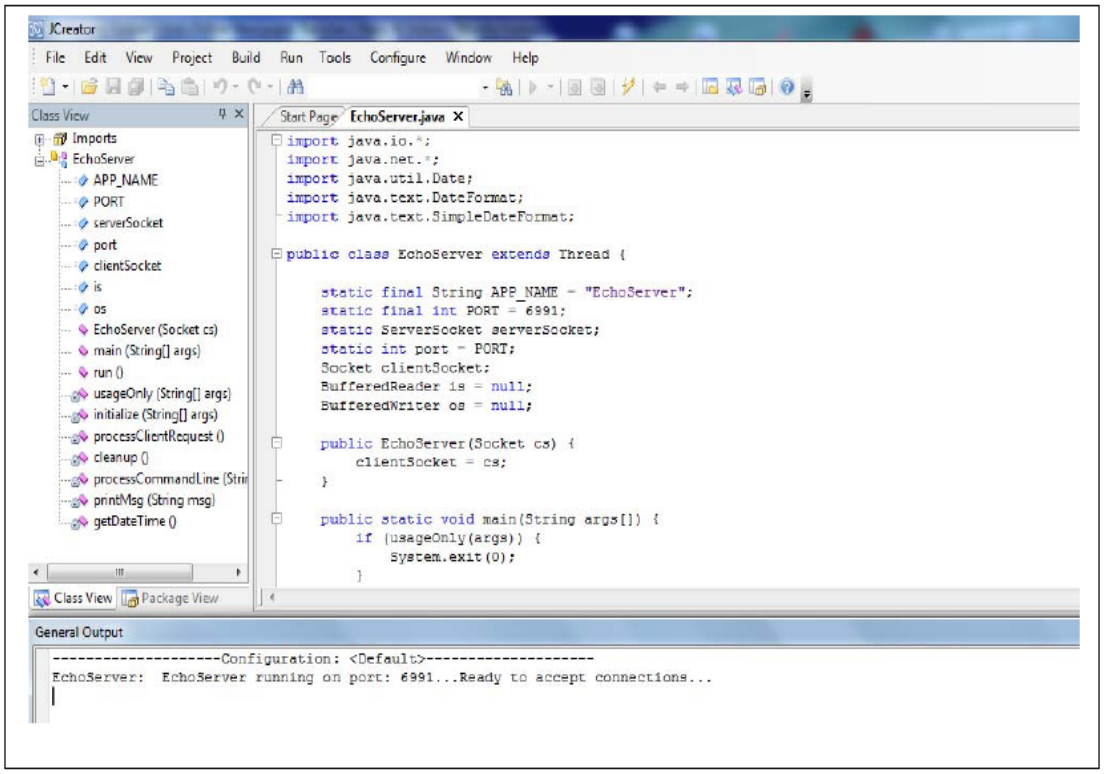


Fig. 6.2.5 Java Echo-Server Application Output

```
import java.net.*;
import java.io.*;
public class EchoClient
{
```

```
public static void main(String[] args) throws IOException {
if (args.length < 2) {
System.err.println("Usage: java EchoClient <99.253.76.32> <255.255.255.0>");
System.exit(0);}
BufferedReader in = null;
PrintWriter out = null;
BufferedReader fromUser = null;
Socket sock = null;
try {
sock = new Socket(args[0], Integer.parseInt(args[1]));
// set up the necessary communication channels
in = new BufferedReader(new InputStreamReader(sock.getInputStream()));
fromUser = new BufferedReader(new InputStreamReader(System.in));
out = new PrintWriter(sock.getOutputStream(),true);
while (true) {
String line = fromUser.readLine();
if (line == null || line.equals("bye"))
{
out.println("bye");
break;
}
out.println(line);
System.out.println(in.readLine());
}
}
catch (IOException ioe) {
System.err.println(ioe);
}
finally {
if (in != null)
in.close();
if (fromUser != null)
fromUser.close();
if (out != null)
out.close();
if (sock != null)
sock.close();}}}
```

Fig. 6.2.6 Java Echo-Client Application

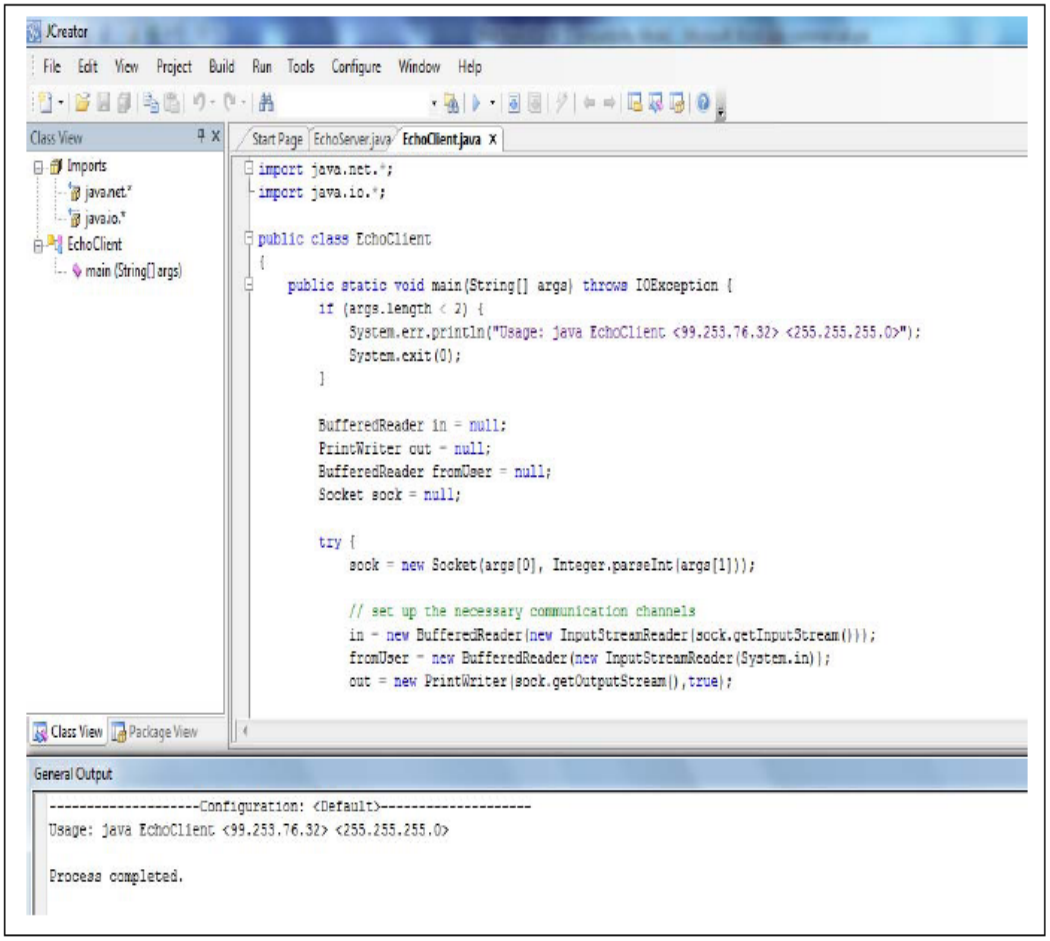


Fig. 6.2.7 Java Echo-Client Application Output

```
import java.util.*;
class getCpuTime{
}import java.lang.management.*;
/** Get CPU time in nanoseconds. */
public long getCpuTime( long[] ids ) {
   ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
   if ( ! bean.isThreadCpuTimeSupported( ) )
      return 0L;
   long time = 0L;
   for ( int i : ids ) {
      long t = bean.getThreadCpuTime( ids[i] );
      if ( t != -1 )
         time += t;
   } return time; }
/** Get user time in nanoseconds. */
public long getUserTime( long[] ids ) {
   ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
   if ( ! bean.isThreadCpuTimeSupported( ) )
```

```
      return 0L;
   long time = 0L;
   for ( int i : ids ) {
      long t = bean.getThreadUserTime( ids[i] );
      if ( t != -1 )
         time += t;
   }
   return time;
}
/** Get system time in nanoseconds. */
public long getSystemTime( long[] ids ) {
   ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
   if ( ! bean.isThreadCpuTimeSupported( ) )
      return 0L;
   long time = 0L;
   for ( int i : ids ) {
      long tc = bean.getThreadCpuTime(  ids[i] );
      long tu = bean.getThreadUserTime( ids[i] );
      if ( tc != -1 && tu != -1 )
         time += (tc - tu);
   } return time;

int repeat = 2000;
                double[] arr = { Double.MAX_VALUE, -3.14e-200D,
Double.NEGATIVE_INFINITY, 567.89023D, 123e199D, -0.000456D, -1.234D, 1e55D
};
                long[] arr2 = { 2283911683699007717L, -8007630872066909262L,
4536503365853551745L, 548519563869L, 45L, Long.MAX_VALUE, 1L, -9999L,
7661314123L, 0L };
                long time;
                StringBuffer s = new StringBuffer();
                Hashtable<Object, Object> h = new Hashtable<Object, Object>();
                System.out.println("Starting test");
                time = System.currentTimeMillis();
                for (int i = repeat; i > 0; i--)
                {
                        s.setLength(0);
                        for (int j = arr.length - 1; j >= 0; j--)
                        {
                                s.append(arr[j]);
                                h.put(new Double(arr[j]), Boolean.TRUE);
                        }
                        for (int j = arr2.length - 1; j >= 0; j--)
                        {
                                s.append(arr2[j]);
                                h.put(new Long(arr2[j]), Boolean.FAarr2E);}}
                time = System.currentTimeMillis() - time;
                System.out.println(" The test took " + time + " milliseconds");}}
```

Fig. 6.2.8 Java CPU Time Application

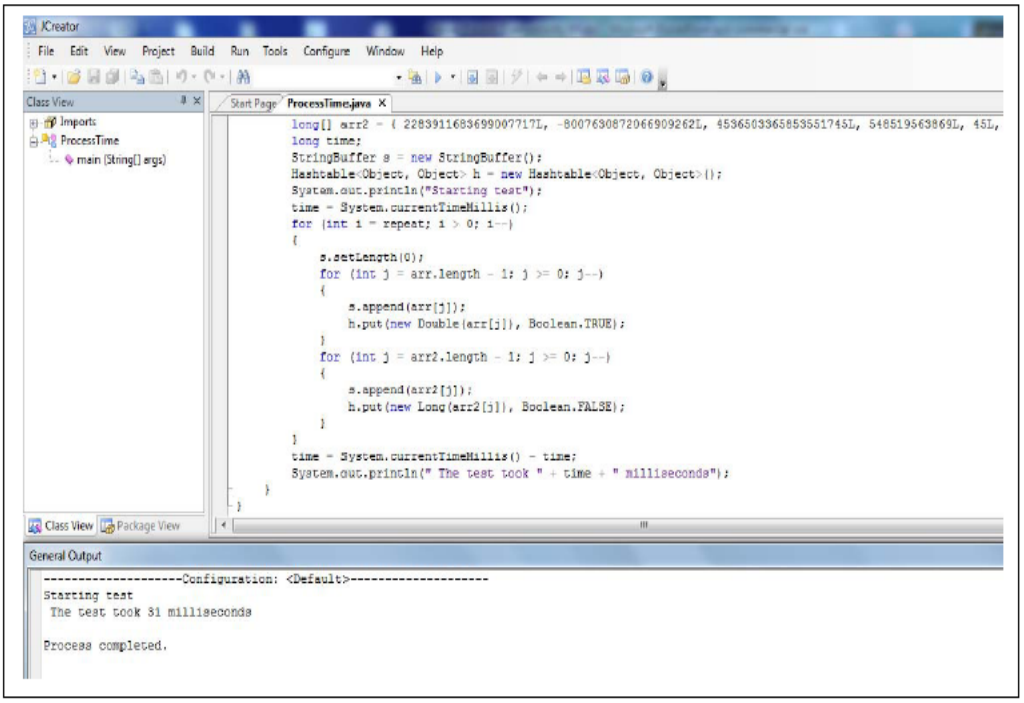


Fig. 6.2.9 Accumulated Delay for Latency and context switching

In parallel programmed client/server, multithreaded socket server application accepts multiple client connections create threads to service every new client. A socket server identifies each client connection with a unique ID through its IP address. For any residual thread, public void flush() flushes the stream. This is done by writing any buffered output bytes to the underlying output stream and then flushing that stream. The public interface flushable is a destination of the data that can be flushed. Socket server on receiving quit command quits residual threads. Normally running both server and client on the same system, gets 0 second because the communication is less than 1 ms.

For a real time system, one can measure the delay by System.nanoTime() command to measure delay in nano seconds.

The graphical representation in Fig. 6.2.10 shows the parent thread which is the main program delay with two child threads delay. The program threads occupying the CPU cycle release the resources and the CPU cycle time seems to go in negative, it is called the wrap around time to reset systems resources for the next run. The time in nano seconds appear as zero seconds.

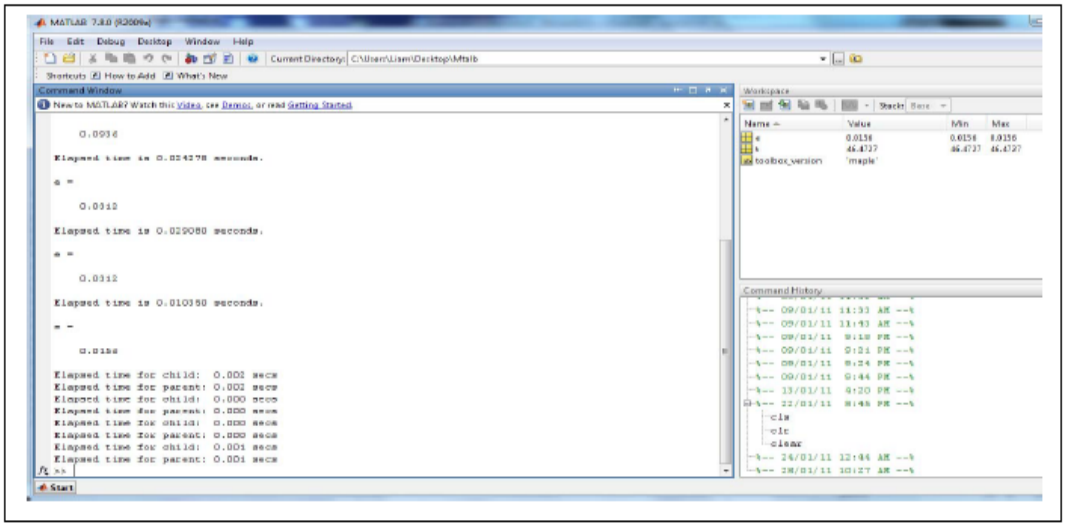


Fig. 6.2.10 MatLab Delay for Parent Child threads and Elapsed time

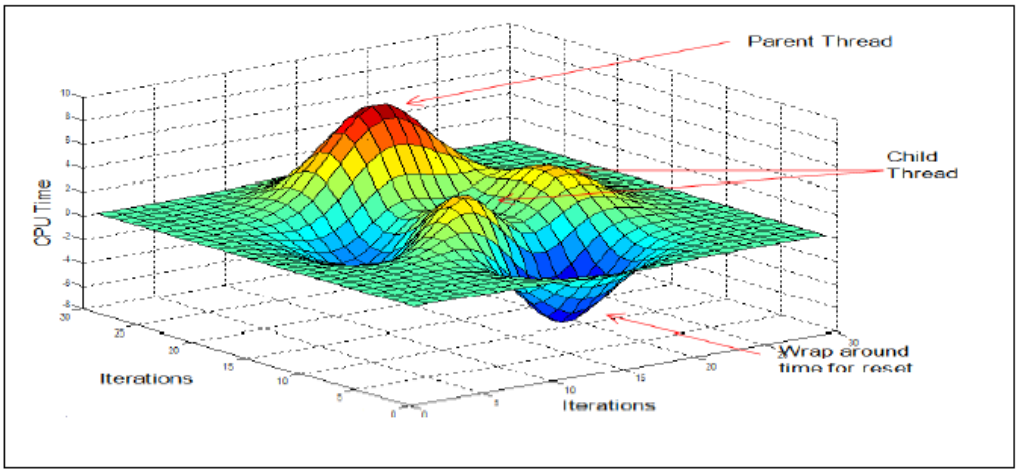


Fig. 6.2.11 MatLab Delay for Parent Child threads and Wrap around time

6.3 Concurrent Programming Logic

The logic behind the concurrent programming model is based on Flynn's taxonomy classification. The single program single data model is omitted as it does not satisfy the basic parallelized program model. The three logics for sensor, controller and actuator model are:

- SPMD = Single Program Multiple Data
- MPMD = Multiple Program Multiple Data
- MPSD = Multiple Program Single Data

The logics are defined below:

If Logic = SPMD

Initialize Sensor1 Data, Sensor 2 Data

Initialize Actuator1 Data, Actuator 2 Data

Start thread

Program (try, catch)

Output result

If Logic = MPSD

Initialize Sensor Data

Start thread

Program1 (try, catch)

Program2 (try, catch)

Program3 (try, catch)

Program4 (try, catch)

Output result

The study implemented the MPMD logic and plot the delays of MPMD for three programs i.e. program 1 with 10 threads, program 2 with 20 threads and program 3 with 40 threads. The delays are shown in micro seconds for a number of iterations. The simulation results show that up to 10000 iterations the delays are much higher for the program 2 and 3 as compared to program 1 delays. For further iterations, the delays of program 1 and program 2 are more or less similar. The output for multithread control network shows that delays of lower order programs (10 to 20 threads) are much lower than delays of higher order programs (40 threads).

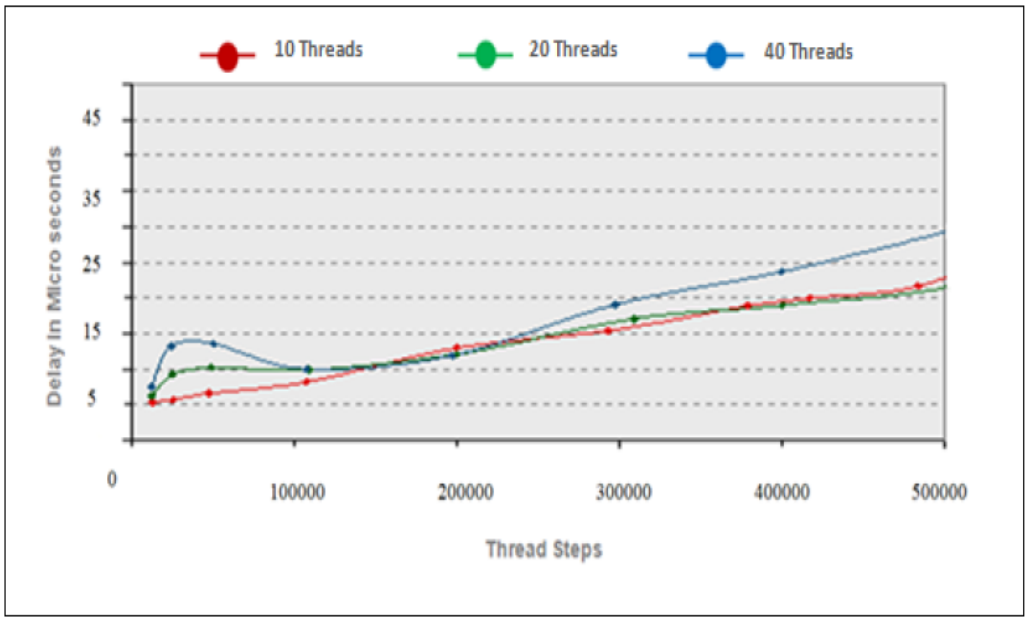


Fig. 6.3.1 MPMD Logic output for a Control Network

For a multi-thread delay, a matlab code was used to show the elapsed time between the threads, and the total time taken by the threads iterations and the time taken by the parent and child threads as shown in Fig. 6.3.1.

As discussed in Chapter three, this study simulated the matlab code to evaluate the delays for fixed cell input queue and virtual output queue for 6x6 FPGA fabric and from the results it is evident that the points for iterative MWM algorithm are concentrated within 0.175 and 0.155 milliseconds; whereas, it is from 0.185 to 0.160 for iterative PIM algorithm. From Fig. 6.3.2 and 6.3.3 it is clear that packet transmission is smoother and average delays for longer packet queues are lesser in iterative greedy MWM than PIM matching. Large packets with higher bandwidth using two class queue model (one link with twice service rate) and iterative MWM transmission can minimize delays.

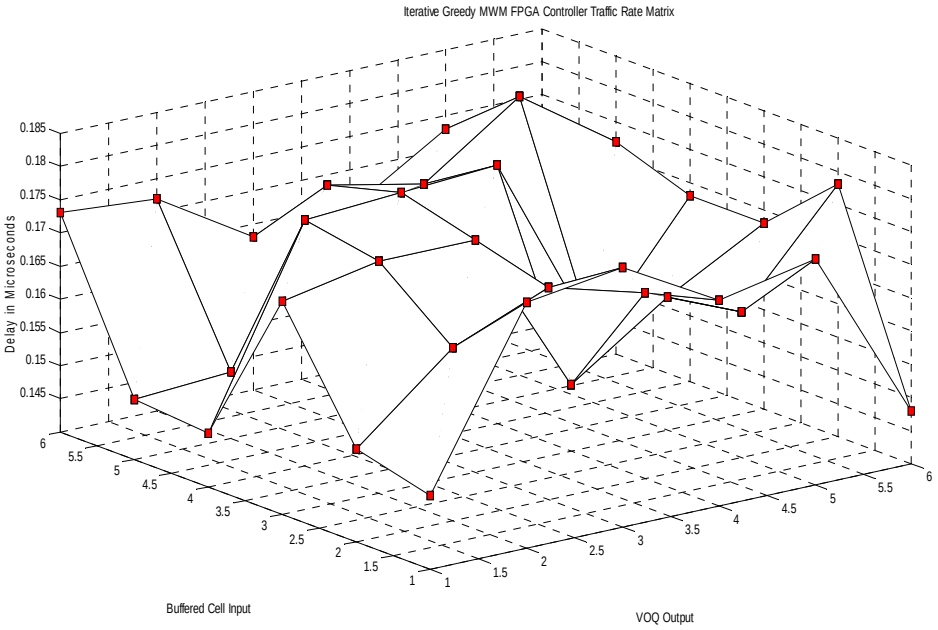


Fig. 6.3.2 Iterative Greedy MWM Matching

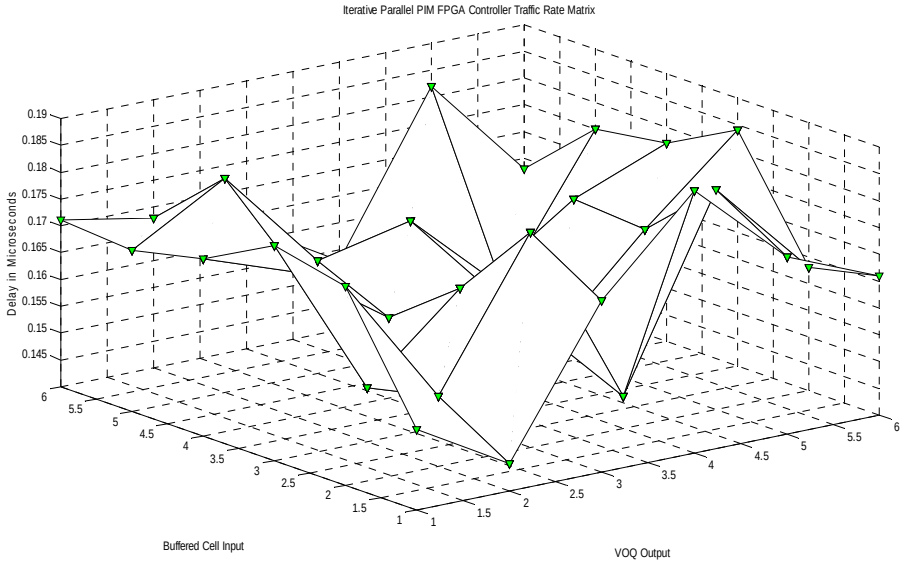


Fig. 6.3.3 Iterative PIM Matching

7. FUTURE WORK

The computations were adapted for a theoretical model of a CANDU SCWR hydrogen co-generation plant. As described below, the study's conclusions are presented followed by recommended areas of future research.

7.1 Conclusions

The study modelled the CANDU SCWR hydrogen communication channel that reviews the wireless protocols, topologies, queuing and scheduling, and client server virtual application to generate communication delays. The study analysed wireless communication protocols, adapted for a single non-pre-emptive and non-prioritized queue. The results show that if one transmits packets within the limitations of the particular protocol specifications, then longer packets, transmitted through higher data rates generate optimized results. The case is also supported by the single input and dual channel queuing model that provides higher service time with fewer delays and higher throughput. Similarly the longer packet assumption is supported by the iterative maximum weight matching, longest queue first strategy. The scheduler selects packets with maximum weight and simulated results show that average delay in MWM virtual output queue is better than parallel iterative matching queue. The results of MWM VOQ only satisfies packets of non-bursty and predictive traffic.

7.2 Recommendations for Future Research

All work was focussed on the mathematical and analytical review of the historical and predictive model. Wireless sensor networks are rarely used in nuclear plants for monitoring and communication. This model computed and simulated values

that can be verified using sensor networks in the co-generation model. Various data accumulation sensors and applications are used in industry, that work on the application layer, media access layer and physical layer of the OSI model. In this section, a review of the technologies available for wireless communication sensor nodes and would show preliminary results that can be focused during future research.

Wireless sensor nodes most popularly known as motes can be simulated using various java based applications that can sense temperature, pressure etc. Contiki and Moteview [53-56] are examples of simulating software that can monitor sensor motes data. Two standard operating systems TinyOS and Contiki are widely used in sensor communication. Sensor motes have built in processor with limited memory (128 kb), transceiver, outsource battery with limited life, and a sensing module. Sensor motes are usually deployed for specific applications. As nuclear power plants safe operation is very sensitive issue, and any failure in system monitoring and control can lead to an accident: therefore, wireless technologies are not implemented in nuclear control rooms and non conventional areas, due to interference with control room communication channels. In this co-generation model an interaction of CANDU SCWR into an oxygen reactor and hydrolysis reactor loop is suggested.

Sensor motes can be used in these loops to monitor parameters such as temperature and pressure. Sensor motes are interrupt driven devices that run on static memory application and are operated using the tiny OS application code, as due to memory scarcity entire operating system cannot be stored on the sensor

mote. Each sensor can sense multiple parameters and can be configured to monitor/control different parameters and compute values on the control station either a laptop or a desktop. The interface is developed between the operating system in a real or a virtual environment between the operating system (Windows Vista, XP, 7 or Mac) and the tinyOS or Contiki. Sensor motes are limited to specific applications and hold limited memory; therefore, First In First Out (FIFO) queuing is the only possible option for multiple packets servicing, although scheduling and queuing delays can be computed for application communicating over a wireless medium between base stations transmitting sensor nodes signals. In real time monitoring, the latency between packets should be bounded by thresholds that define a retransmission policy in the case of packet drop.

Response time between motes and the application should operate in a split phase mode i.e. a send call is issued by sensor mote and upon receiving of the acknowledgement actual packet is transmitted; therefore, the transmission is split into a send call and an actual sending of the packet. In this discussion of the interrupt handlers for hardware and software interface, the motes operate in two thread mode, interrupt thread of higher priority and scheduler thread of lower priority. Whenever a call is issued an interrupt sequence is generated to engage the available resources for parameter signal transmission, that is bounded be a latency delay. Once resources are allocated through a call sequence, the packets containing parameter information are transmitted over a wireless communication

medium via a base station to the computation station housed with a computer in the form of application software and operating system.

This study generated the delay values between the OSI model layers and for a virtual environment that generates more accurate delay values for a virtual client server cluster computing. The future work should be focused on designing an interface to sense parameters using motes and integrating those sensor motes into the existing infrastructure of the hydrogen lab. The sensor mote tinyOS interfaced over a wireless link forms XMesh network, that allows users, over the air programming (OTAP) of the mote note. Fig. 7.2.1 shows the server, gateway and the XMesh network architecture that follows the step phase communication by sending the request and getting the response. The local server is assumed as a virtualized server that can be Windows 2008 hyper v virtual environment or uses the VMware over a Windows environment.

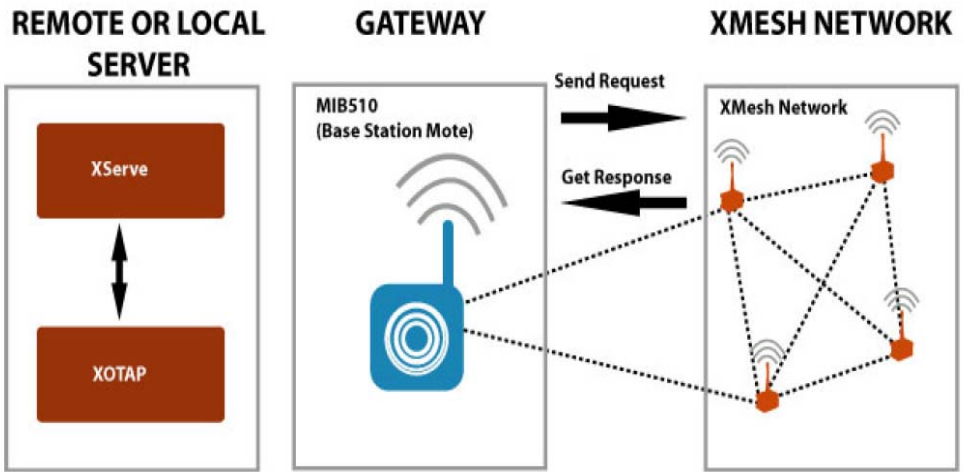


Fig. 7.2.1 Wireless Sensor Network Architecture [42]

There are various simulators used to develop an interface between the sensor nodes and the server, such as Mote View, Mote Works and Cooja. These simulators are simpler to install over a windows virtual environment. When one connects the Mote sensor adapter USB serial port with the computer, and run the setup files, the mote can be detected by the OS on the server and one can place the transmitter over a range of fifty (50) metres to detect the parameters. Fig. 7.2.2 shows the programming module over the air interface for a mote.

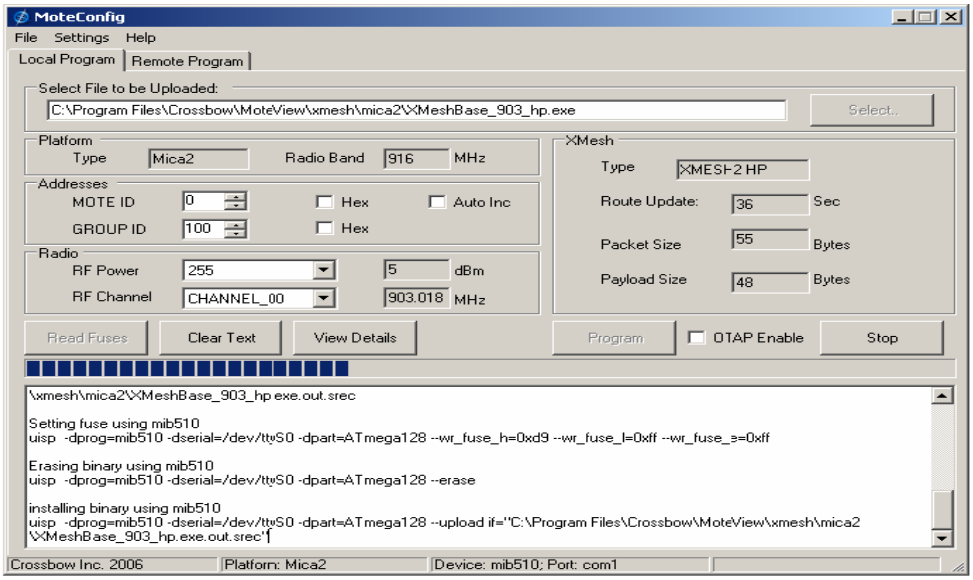


Fig. 7.2.2 OTAP Programming Phase for the Sensor Mote [42]

For queuing analysis, it was assumed with a simpler scenario of dual slot of single packet transmission without pre-emptive priority transmission. For future work, multiple techniques of pre-emptive and non-pre-emptive prioritized queues are recommended. The scheduling model developed in this research shows that

maximum weight matching is better option in non-bursty traffic. These results can be tested in a real environment.

References

[1] L. Grunske, P. A. Lindsay, N. Yatapanage, and K. Winter, "An Automated Failure Mode and Effect Analysis Based on High-Level Design Specification with Behavior Trees," In Proceedings of IFM 2005, volume LNCS 3771, pages 129-149. Springer, 2005.

[2] Cassanelli G., G.M., Fantini F., Vanzi M., B. Plano, "Failure Analysis-assisted FMECA," Microelectronics and Reliability, 46: p. 1795-1799, 2006.

[3] M. H. Mazouni, D. Bied Charreton, J.F. Aubry, "Proposal of a generic methodology to harmonize Preliminary Hazard Analyses for guided transport", IEEE – SMC international conference: system of systems engineering, San Antonio-Texas - USA, April 2007.

[4] P J Wilkinson, T P Kelly, "Functional Hazard Analysis for Highly Integrated Aerospace Systems," IEEE Certification of Ground/Air Systems Seminar, No.1998/255.

[5] Hunt, A., Kelly, B.E., Mullhi, J.S., Lees, F.P., & Rushton, A.G., "The propagation of faults in process plants: 8. Control systems in fault tree synthesis," Reliability Engineering, 39, 211-227, 1993.

[6] Garrett, S., S. Guarro and G. Apostolakis: "The Dynamic Flowgraph Methodology for Assessing the Dependability of Software Systems," *IEEE Transactions on Systems, Man and Cybernetics* 25, pp.824-840, 1995.

[7] Yau, M., S. Guarro and G. Apostolakis: "Demonstration of the Dynamic Flowgraph Methodology using the Titan II Space Launch Vehicle Digital Flight Control Software," *Reliability Engineering and System Safety* 49, pp.335-353, 1995.

[8] Garrett, S., M. Yau, S. Guarro and G. Apostolakis: "Assessing the Dependability of Embedded Software Systems Using the Dynamic Flowgraph Methodology," in *Dependable Computing and Fault-Tolerant Systems Vol. 9*, F. Cristian, G. Le Lann, T. Lunt (eds.), Springer-Verlag Wien, New York, 1995.

[9] Houtermans M., G. Apostolakis, A. Brombacher and D. Karydas. "Programmable electronic system design & verification utilizing DFM." In *SAFECOMP 2000* (F. Koornneef and M. van der Meulen, eds.), Lecture Notes in Computer Science 1943 (2000), 275-285.

[10] Ahmad W. Al-Dabbagh and Lixuan Lu, **"**Reliability modeling of networked control systems using dynamic flowgraph methodology", Reliability Engineering & System Safety, Volume 95, Issue 11, November 2010, Pages 1202-1209.

[11] Ahmad W. Al-Dabbagh, Lixuan Lu, and Mazza Antonio, "Modelling, simulation and control of a proton exchange membrane fuel cell (PEMFC) power system," International Conference of Hydrogen Production (ICH2P-09), Oshawa , CANADA, 2010, vol. 35, pp. 5061-5069.

[12] Ahmad W. Al-Dabbagh and Lixuan Lu, "Dynamic flowgraph modeling of process and control systems of a nuclear-based hydrogen production plant," vol. 35, pp. 9569-9580 International Journal of Hydrogen Energy, 2010.

[13] Ahmad Wail Al-Dabbagh, "Dynamic flowgraph methodology for reliability modelling of networked control systems: with application to a nuclear-based hydrogen production plant," MASc Thesis Electrical and Computer Engineering, UOIT, Oshawa, 2009.

[14] www.ascanic.com Accessed at May 14, 2011.

[15] Hesham El-Rewini, Mostafa Abd-El-Barr, "Advanced Computer Architecture and Parallel," A John Wiley & Sons, Inc Publication, 2005.

[16] Thomas, M.S.; Kumar, P.; Chandna, K., "Design, development, and commissioning of a supervisory control and data acquisition (SCADA) laboratory for research and training," IEEE Transactions on Power Systems publication information, Volume: 19, Issue:3, pp 1582 – 1588, Aug. 2004.

[17] Fayyaz Ahmed, Rashid Faruqui, Mudassar Imran, "Collaborating techniques of CDMA nodes," IEEE Multitopic Conference, 2008. INMIC, IEEE International pp 278 – 285, 23-24 Dec. 2008.

[18] James R. Moyne and Dawn M. Tilbury, "The Emergence of Industrial Control Networks for Manufacturing Control, Diagnostics, and Safety Data," Proceedings of the IEEE, Vol. 95, No. 1, January 2007.

[19] RTEMS Filesystem Design Guide, Edition 4.10.99.0, for RTEMS 4.10.99.0, 27 June 2007.

[20] John A. Stankovic and R. Rajkumar, "Real Time Operating Systems", Real Time Systems, 28, 237-253, 2004.

[21] T. Straumann, "Open Source Real Time Operating Systems Overview," 8th International Conference on Accelerator & Large Experimental Physics Control Systems, 2001, San Jose, California

[22] S. Gorinsky, S. Jain, H. Vin, and Yongguang Zhang, "Design of Multicast Protocols Robust against Inflated Subscription", *IEEE/ACM Transactions on Networking*, Vol. 14, No. 2, pp. 249-262, Apr 2006.

[23] Yu-Doo Kim and Il-Young Moon, “Improved AODV routing protocol for wireless sensor network based on ZigBee”, In Proceedings of IEEE Advanced Communication Technology Conference , 2009.

[24] Majid S. Naghmash, Mohd Fadzil Ain, and Chye Yin Hui, “FPGA Implementation of Software Defined Radio Model based 16QAM”, European Journal of Scientific Research, Vol.35 No.2 (2009), pp 301-310.

[25] Nachiondo Teresa, lich Jose, and Duato Jose, “Destination-based HoL blocking elimination Destination-based HoL blocking elimination”, In Proceedings of the 12th International Conference on Parallel and Distributed Systems, 2006.

[26] Mark W. Goudreau, Stavros G. Kolliopoulos, and Satish B. Rao, "Scheduling Algorithms for Input-Queued Switches: Randomized Techniques and Experimental Evaluation", In *Proceedings of IEEE INFOCOM*, pp 1634–1643, 2000.

[27] Dong-Gi Lee, Gwang-Hee Heo, and Jong-Myung Woo, “ New Bounds using the Solution of the Discrete Lyapunov Matrix Equation”, International Journal of Control, Automation, and Systems Vol. 1, No. 4, December 2003.

[28] Ben Charrada, Faouzi, Ezouaoui, Sana and Mahjoub, Zaher , “Greedy algorithms for optimal computing of matrix chain products involving square dense and triangular matrices”, January 2011 - Volume 45, Issue 01 International Journal, RAIRO - Operations Research.

[29] W. E. Vesely, et. Al., “Fault Tree Handbook”, Systems and Reliability Research, Office of Nuclear Regulatory Research, January 1981. NUREG 0492.

[30] Garrett, S., S. Guarro and G. Apostolakis: “The Dynamic Flowgraph Methodology for Assessing the Dependability of Software Systems,” *IEEE Transactions on Systems, Man and Cybernetics* 25, pp.824-840, 1995.

[31] H. Khartabil, “SCWR: Overview,” GIF Symposium, 9-10September 2009.

[32] S. S. Bae Et. Al., “Status of Ongoing Research on SCWR Thermal Hydraulics and Safety,” GIF Symposium, Paris, France, 9-10 September, 2009.

[33] Davit Danielyan, “Super Critical Water Cooled (SCWR) Reactor System-as One of the Most Promising Type of Generation IV Nuclear Reactor Systems,” Nov. 4, 2003.(http://www.tkk.fi/Units/AES/courses/crspages/Tfy 56.181_03/Danielyan.pdf) Accessed at June 2010.

[34] Baindur, S. “Materials Challenges for the Super Critical Water-Cooled Reactor (SCWR),” Bulletin of the Canadian Nuclear Society, Vol. 29 No. 1, March 2008, pp. 32-38.

[35] J. Buongiorno and P.E. MacDonald, “Supercritical Water Reactor (SCWR) Progress Report” for 2003 Generation-IV R&D Activities for the Development of the SCWR in the U.S., INEEL, September 30, 2003.

[36] International NERI (I-NERI) projects Report “Appendix 2.0 Supercritical Water Reactor,”http://nuclear.inl.gov/deliverables/docs/appendix_2.pdf (Accessed at May 2010)

[37] Naidan et, Al. “SCW NPPs: Layouts and Thermodynamic Cycles, “Proceedings of the International Conference Nuclear Energy for New Europe, Bled, Slovenia, Sept. 14-17, 2009.

[38] Duffey, R.B., et. Al., “Supercritical Water-Cooled Nuclear Reactors (SCWRs): Current and Future Concepts - Steam-Cycle Options”, Proc. ICONE-16, Orlando, FL, USA, May 11-15, Paper #48869, 2008, 9 pages.

[39] Pioro, I.L. and Duffey, R.B.,. Heat Transfer and Hydraulic Resistance at Supercritical Pressures in Power Engineering Applications, ASME Press, New York, NY, USA, 2007, 334 pages.

[40] Main Steam Supply and Feed water System, http://canteach.candu.org/library/19930205.pdf (Accessed at January December 2009).

[41] Naterer G.F. et. Al., “Recent Canadian advances in nuclear-based hydrogen production and the thermo chemical Cu–Cl cycle,” International Journal of Hydrogen Energy 34 (2009) 2901- 2917.

[42] Naterer, G. F. et. Al. , ``Canada's Program on Nuclear Hydrogen Production and the Thermochemical Cu-Cl Cycle'', International Journal of Hydrogen Energy, vol. 35, pp. 10905 - 10926, 2010

[43] Naterer, G. F., et. Al., ``Synergistic Roles of Off-peak Electrolysis and Thermochemical Production of Hydrogen from Nuclear Energy in Canada'', International Journal of Hydrogen Energy, vol. 33, pp. 6849 - 6857, 2008.

[44] P. Chatzimisios, A. C. Boucouvalas and V.Vitsas, “IEEE 802.11 Packet Delay – A Finite Retry Limit Analysis”, In Proceedings of IEEE GLOBECOM 2003, pp 950-954.

[45] Haitao Wu, Shiduan Cheng, Yong Peng, Keping Long, and Jian Ma, "IEEE 802.11 Distributed Coordination Function (DCF): Analysis and Enhancement", Journal of Computer Science and Technology, Volume 18, Number 5, 607-614.

[46] P. Raptis, V. Vitsas, K. Paparrizos, P. Chatzimisios, A. C. Boucouvalas, and P. Adamidis, "Packet Delay Modeling of IEEE 802.11 Wireless LANs", Journal of Mobile Networks and Applications, Volume 14, Issue 6, December 2009.

[47] Jasmin Velagic, "Design of Smith-like Predictive Controller with Communication Delay Adaptation", World Academy of Science, Engineering and Technology 47, April 2008.

[48] Ivan N. Vukovic, "Delay Analysis of Different Backoff Algorithms in IEEE 802.11," IEEE Vehicular Technology Conference, IEEE 60th Conference Sept. 2004 pp 4553 - 4557 Vol. 6.

[49] Zhijia Chen, Chuang Lin,P Hao Wen, and Hao Yin, "An Analytical Model for Evaluating IEEE 802.15.4 CSMA/CA protocol in Low-rate wireless application", In Proceedings of 21st International Conference on Advanced Information Networking and Applications Workshops, 2007.

[50] Petr Jurčík, Anis Koubâa1,, Mário Alves, Eduardo Tovar, and Zdeněk Hanzálek, "A Simulation Model for the IEEE 802.15.4 Protocol: Delay/Throughput Evaluation of the GTS Mechanism", In Proceedings of the 15th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). October, 2007.

[51] Serialized IRQ Support for PCI Systems, Revision 6.0, Compaq Computer Corporation, Cirrus Logic Incorporated, National Semiconductor Corporation, OPTI Incorporated, Standard Microsystems Corporation, Texas Instruments Incorporated, VLSI Technology Incorporated, September, 1995. Accessed at January 28, 2011.

[52] www.jcreator.com, Accessed at February 7, 2011.

[53] Marcus Lunden and Adam Dunkels, "The Politecast Communication Primitive for Low-Power wirelesses," The ACM SIGCOMM Computer Communications Review, April 2011.

[54] Jeonggil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Andreas Terzis, Adam Dunkels, and David Culler, "ContikiRPL and TinyRPL: Happy Together", In *Proceedings of the IP+SN 2011 workshop*.

[55]Adam Dunkels, Luca Mottola, Nicolas Tsiftes, Fredrik Österlind, Joakim Eriksson, and Niclas Finne,"The Announcement Layer: Beacon Coordination for the Sensornet Stack",. In *Proceedings of EWSN 2011*.

[56] Alec Woo, Siddharth Seth, Tim Olson, and Jie Liu, “A Spreadsheet Approach to Programming and Managing Sensor Networks”, *IPSN’06,* April 19–21, 2006, Nashville, Tennessee, USA.