Real-Time Ethernet Networks Simulation Model

Taweewit Pensawat

December 21, 2006

School of Information Science, Computer and Electrical Engineering,
Halmstad University, Box 823, S-301 18 Halmstad, Sweden
Email: taweewit.p@siamu.ac.th, Tel: +46-737287618

Contents

1 Abstract 3
2 Objective 3
3 Audience 3
4 Introduction 4
5 Problem Statement 5
6 Project Limitation 6
7 Related Technologies 7
  7.1 Ethernet 7
  7.1.1 Ethernet Frame 7
  7.1.2 CDMA/CD 7
  7.1.3 Full Duplex 7
  7.2 Available Ethernet for Industrial Automation 8
  7.2.1 Ethernet/IP 8
  7.2.2 EtherCAT 9
  7.2.3 IEEE 1588 9
  7.2.4 RETHER 10
  7.3 Switched Real-Time Ethernet with Earliest Deadline First
     Scheduling 10
8 Real–Time communication background 11
  8.1 Real–Time Systems 11
  8.2 Aperiodic or Sporadic System 12
1 Abstract

Real-time networks are traditionally built on proprietary standards, resulting in interoperability issues between different real-time network implementations and traditional data networks mainly used in back office operations. Continuity and supplier independence are a cause of concern with current proprietary real-time networks.

This project evaluates the capability of providing real-time traffic over switched Ethernet with EDF scheduling algorithm implemented at both the switch and the node. By using OMNET simulation tool at packet level, it is shown that the EDF implementation in switched Ethernet can guarantee real-time traffic over the network and at the same time supporting non real-time traffic.

2 Objective

Ascertaining the feasibility degree of real-time Ethernet networks in environments requiring real-time communication by simulating it with comparable network utilization characteristics found in traditional non-Ethernet real-time networks and studying these results to find out if a real-time Ethernet network could effectively substitute a proprietary real-time network.

3 Audience

This paper encourages Ethernet network device manufacturers and proprietary real-time parties to consider entering the real-time Ethernet market based on findings presented this paper, although real-time Ethernet is upcoming, the maturity of the products is unsuitable for live deployments at this time. Further improvements in reliability of real-time Ethernet devices and industry adoption is necessary.
4 Introduction

Real-Time communication is a major development in industrial automation and computer science. It is widely implemented and can be seen in various applications such as flight control, manufacturing line, health care operation and other applications where accurate timing is required. In order to maintain time constraint of those systems only possible with real-time communications, network devices, protocols and applications must be specifically designed to meet requirements of real-time communications.

The trend to lower supplier dependency and interoperability issues could only be archived by augmenting existing network standards that are widely adopted so that it can handle real-time traffic amongst other types of traffic. The demand for suitable alternatives is increasing as typical mixed-mode installations are subject to high maintenance costs due to double industry standards requiring more engineering resources to service them.

Ethernet is the most adopted local area network available today[3], Typical to our case it is mostly utilized in the back office to overview real-time networks, interconnection with real-time networks is accommodated with specialized bridges responsible for converting Ethernet traffic to proprietary real-time networks.

Therefore the candidate platform for efficient real-time communications is Ethernet, by itself however, does not provide any support for real-time traffic because of CSMA/CD (Carrier Sense Multiple Access with Collision Detection) which is used by Ethernet to resolve collision problems in a non-deterministic manner which leads to unpredictable initiating time and latency to destination will be uncertain as well.

Switching technology, a major improvement in the last 10 years, widely adopted today, was developed to enhance the speed of networks and increase the efficiency, resulting in faster networks over traditional Ethernet networks utilizing hubs. Collisions in a fully switched network are drastically reduced as the traffic is repeated only to known recipients residing on specific ports according to the internal layer-2 routing table maintained by the switch.

Although the possibility of collisions are strongly reduced but not eliminated, the Collision Detection(CD) routine is still required to prevent port flooding since single ports could receive data from multiple ports simultaneously creating congestion at the receiving port, to report such collisions to the sending port, the Collision Detection routine is required and will report congestion so that the host sending the packets will delay sending more packets until the congestion subsides.

The Collision detection routine is mandatory in Ethernet networks, however it could adversely affect the strict priority of real-time packets, therefore it is important in real-time Ethernet networks that typical collision detection is prevented from being applied to real-time packets at any cost since it could unregulated the timing in an unpredictable way.
Still there are obstacles that need to be crossed. In a traditional Ethernet networks, a collection of different network replicating equipment could be interconnected and could be manufactured by different suppliers introducing additional variables due to different traffic handling characteristics, which would be the usual case in larger Ethernet networks, varying packet handling latency across different parts of the network will negatively affect real-time traffic. Although this is outside the scope of this paper, these issues need further study when real-time Ethernet is considered for implementation in such a live environment.

5 Problem Statement

A single and open standard network system is needed in industrial automation because there is strong demand to reduce the cost of investment in networking and application development while such a network could deliver much more value added features to the network, besides core features mainly presented in this paper.

Fieldbus systems and other proprietary real-time network solutions could not deliver such breakthrough in the near future because they have limited in-house human resources and opensourcing would not be possible because of legal concerns.

Ethernet has been serving general purpose local area networks for decades and its openness of standard is driving down cost of Ethernet products to the level that even individuals could afford Ethernet networks.

Fully switched network seems to be the answer and could be used in office automation. Because modern Ethernet switches work with new technologies enabling full-duplex communication. This results in increasing bandwidth allowances and at the same time reducing collisions as well as latency, in general it could be said that the overall traffic handling characteristics of modern Ethernet switches is very good. However, the first-come-first-serve scheduling algorithm which stems from Ethernet remains the same. On an Ethernet node, Packets will be sent out of the NIC on a first-come-first-serve basis depending on which application request arrives to the operating systems kernel first. On an Ethernet switch, the same thing happens but this time it is dependent on which packet arrives first. Such behavior of Ethernet does not benefit real-time communications where packets must arrive at destinations within strict time constraints because missing deadlines means severe damages to operations inline. Many approaches have been introduced in the last decade to make Ethernet support real-time communication, one of them is more interesting to us than the others: Adding a real-time layer to traditional OSI model without modifying hardware both at the node and Ethernet switch. The Real-Time layer is inserted between the transport layer and network layer, it works by sorting packets in the queue using Ear-
liest Deadline First (EDF) algorithm to achieve real-time communications. However, implementing such an approach into Ethernet could be a time-consuming task and cost of development will be very high since developing time must be spent on coding with low or mid-low level programming language on clients and also low-level programming on the Ethernet switch. To solve this problem, simulation technique are chosen to study the feasibility of how EDF works in Ethernet networks. It has been used for decades especially in computer science and engineering since simulation in this manner is essentially scientific proof of accurateness.

6 Project Limitation

Limited resources are allocated to this project, therefore certain limits have been set to reduce the variables throughout the course of the project. Care has been taken to set reasonable limitations that are comparable to real live scenarios.

All simulation models are built on these limitations.

- The simulations are conducted at packet level only. It can be assumed that the packet is UDP, or otherwise connectionless unlike TCP.

- There are 2 types of nodes: Real-time nodes and non real-time nodes. No single node generates both real-time and non real–time traffic.

- Every node including the switch operates only in full-duplex mode.

- There is only one Ethernet switch.

- Propagation delay and frame processing time at the switch is fixed. Adjusting is possible in hard–code.

- System overheads such as context-switching time are not considered.

- No aperiodic task are tested.

- All processes have fixed execution time but can be changed easily from configuration file.

- This simulation model is made on Cygwin 1.5.21 for Windows (Details in Software section). Porting the model to other operation systems is possible but re–compiling is needed and instructions are not covered in this paper.

- It is assumed that there is only one system processor at each node.

- Real–Time Operating Systems are not considered in the project.
7 Related Technologies

7.1 Ethernet

Ethernet is a frame based communication system which carry digital signals to distributed stations or nodes. Stations, normally Personal Computers, are connected on the shared medium called Ether. Ether itself is a passive medium which no master station as they are connected in a loosely coupled fashion.

Even Ethernet originated from Xerox’s Palo Alto Research Center with joint development from Xerox, Intel and DEC but the term “Ethernet” nowadays frequently refers to IEEE 802.3 standard, the IEEE code that will be further used to refer Ethernet in this paper.

There are 3 basic components in Ethernet system. First, physical medium which is responsible for sending electric signals to medium. Second, medium access control which enabling the sharing media. Third, Ethernet frame which consist of sequence of bits.

7.1.1 Ethernet Frame

<table>
<thead>
<tr>
<th>Filed length in byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 1 6 6 2 46-1500 4</td>
</tr>
<tr>
<td>Preamble  SOF  Destination Address  Source Address  Type  Data  FCS</td>
</tr>
</tbody>
</table>

Figure 1. Ethernet 802.3 Frame

7.1.2 CDMA/CD

Ethernet uses CDMA/CD as medium access control mechanism. With this mechanism, Ethernet is capable of detecting if there is transmission on the medium with Carrier Sense(CS), nodes have the same priority to compete to get into medium with Multiple Access(MA) and when collision occurs, sending nodes will stop and random starting time again with Collision Detection(CD)

7.1.3 Full Duplex

As the name implies, full duplex means nodes can transmit and receive data at the same time. It is a feature that comes with Ethernet switches enabling bi-directional data transfer between 2 nodes, effectively doubling the data transfer.
7 RELATED TECHNOLOGIES

7.2 Available Ethernet for Industrial Automation

Ethernet in industrial automation is not something new but it has been around for years. Each proprietary product has its own strength and technique implemented into Ethernet to make it supports real-time communications. This sections will give brief explanation of available Ethernet solutions to serve real-time communication to industrial automation in the market.

7.2.1 Ethernet/IP

Ethernet/IP is a producer/consumer network protocol for industrial automation which is developed by ControlNet International, Open DeviceNet Vendors Association(ODVA) and Industrial Ethernet Association(IEA). The OSI 7-Layer is used as a framework for Ethernet/IP as well as other network protocols. Ethernet/IP implements Common Industrial Protocol(CIP), it is implemented to combine top 3 layers into one as shown on figure 2.

CIP plays significant role in Ethernet/IP as a protocol that defines data exchange at the I/O level via explicit messages.

In Ethernet/IP, both TCP and UDP are available. TCP is used to send service information to nodes because such service does not concern about time constraint. For real-time communication, Ethernet/IP uses UDP in unicast or multicast mode. When sending, Ethernet/IP’s frames contain only data because service information was sent earlier with TCP.

Ethernet/IP is compatible with other 2 open network standards: ControlNet and DeviceNet in term of software layer and hardware interface, those 2 standards are also under maintenance of ODVA.

![Figure 2. Ethernet/IP Stack](image)

However, Ethernet/IP plus CIP is not a real-time protocol by itself but with help of CIPSync which is the protocol based on IEEE 1588.

Ethernet/IP uses both UDP and TCP in communication over IP Network but for real-time communication it uses UDP/IP.
7.2.2 EtherCAT

EtherCAT stands for Ethernet for Control Automation Technology. It was developed by a German company, Beckhoff Automation GmbH. It offers determinism from its proprietary OS at master node and Y driver at end node.

EtherCAT encapsulates its data, header and Working counter (WC) field into the data field of standard Ethernet field.

![EtherCAT Diagram](image)

Figure 3. EtherCAT

7.2.3 IEEE 1588

IEEE 1588, also known as Standard Precision Time Protocol, standard is a protocol that defines procedures to assure that real-time data will be synchronized with high accuracy in Ethernet networks. It works in conjunction with other industrial solutions such as Ethernet/IP, EtherCAT and PROFINet to help them deliver real-time communication in a true sense.

A single master clock in the network will synchronizes all slave clocks. Actually, a clock can perform both roles but only one role at a time.

Selection of master clock is done with the Master Clock Algorithm, to synchronize clocks, the master clock sends 2 synchronization packets to its slave for offset.

Selection of master clock is done with the Master Clock Algorithm, to synchronize clocks, the master clock sends 2 synchronization packets to its slaves for offset correction, the next phase is to calculate propagation delay.
There is another time protocol called Network Time Protocol or NTP which is well-known to Internet users but it as designed based on peer-to-peer fashion and suitable for synchronization of remote clocks mainly across WAN, while IEEE 1588 was designed for Ethernet in office automation, but not restrict to and can be used in networks where multicast is implemented.

### 7.2.4 RETHER

REETHER is the project that offers a latency guarantee mechanism solely from the software part without hardware modification. It differs from solutions previously mentioned because this single side requirement. And if compared with our simulation model, readers can spot easily that they are quite similar in terms of concept. RETHER offers real-time communication by modifying lowest level of software layer only. Moreover, RETHER is transparent to an Ethernet switch or inter-networking device.

REETHER has 2 modes of operation namely the CSMA/CD mode for non real-time traffic and RETHER mode for real–time traffic.

When there is no real–time traffic, the network will be running in normal CSMA/CD mode. Every node has the same priority to acquire rights to the medium but when a node, hereafter called initiator, needs to create a real-time channel, the node will send out a special frame called Switch-to-REETHER to tell every node in the same broadcast domain. After all running nodes receive the frame, they will respond by putting outgoing frames on hold and complete outgoing frames in the local buffer. When they finish sending those frames, nodes collectively send an acknowledgment to initiator. The network operation mode changes to RETHER and the time token protocol takes over.

### 7.3 Switched Real-Time Ethernet with Earliest Deadline First Scheduling

Hoai Hoang, Magnus Jonsson, Ulrik Hagstrom and Anders Kallerdahl proposed the method that guarantees delay of real–time traffic in fully switched network without modifying hardware in both the switch and node. The real-time layer, real–time layer, is inserted to OSI model to sort packets before leaving interfaces. Real–time layer sort packets according to their deadline using Earliest Deadline First algorithm.

They have done simulation test in admission control to prove their idea and performance of the approach is shown in graphical format.

This work is different from solutions mentioned earlier because all of them are commercial products and have been available in the market while this study is at its initial stages.

Our simulation model on packet level follows this work with the same base idea to add real–time layer and EDF algorithm to sort packets in order.
to guarantee delays for real-time traffic.

8 Real-Time communication background

8.1 Real-Time Systems

Oxford Dictionary of Computing explains the meaning of real-time system:
"Any system in which the time at which output is produced is significant. This is usually because the input corresponds to some movements in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.”

From the explanation, it can say that real-time systems are the systems that have time constraints and the correctness of real-time systems do not depends only on logical results but with also time or timeliness.

How fast of real-time systems sometimes confuses people because some people think that they must be fast which is only partly correct. For example, airbag control system may need very fast response time in microseconds while missile guidance system may need response time in seconds and both them still real-time systems.

Real-time systems fall into 2 main categories which are hard real-time and soft real-time.

**Hard Real-Time**

For hard real-time, deadlines must be met otherwise severe consequences occur. To guarantee that all deadlines will be met, it has to know worse case execution time and system also has to be able to forecast if there is any job misses deadline.

Hard real-time is a vital part of many computing and control systems today. Here are examples of hard real-time systems.

- Air traffic control system
- Nuclear Power plants
- Robotics
- Operation Systems
- Industrial Automation

**Soft Real-Time**

Soft real-time normally can be seen in applications or operating systems. The systems or applications with soft real-time implemented will try to meet deadlines as much as possible but lagging to meet the deadline does not cause any severe damages but business losses in some cases.

Examples of soft real-time systems are
- Video and Audio Streaming
- Voice over Internet Protocol (VoIP)
- Automatic Teller Machine (ATM)

### 8.2 Aperiodic or Sporadic System

In a aperiodic system, time to release a job is unpredictable such as when the computer user hits a keystroke. In this system, job is released at a time $r_i$ and characterized by a worst case execution time $c_i$. Absolute deadline is $d_i$, which implies that the job has to be executed before then.

$$\text{Relative Deadline } D_i = d_i - r_i$$

### 8.3 Periodic System

Most real-time systems produce periodic tasks such as signal generated from sensors or actuators to a processing unit at end node. To guarantee that each of every job meets its deadline, real-time scheduling algorithm is the algorithm that will be used. This process is called feasibility test. Non-periodic tasks are special tasks with each job inside is independent which can be produced in manufacturing line as well but we will not discuss or study those kinds of tasks in this paper.

In periodic systems, each task is characterized by these parameters

1. $T_i$ = invocation period
2. $c_i$ = worst case execution time (WCET)
3. $D_i$ = relative deadline. Usually equal to $T_i$

Figure 4 shows characteristics of periodic systems in graphical format.

![Figure 4. A periodic system.](image)

General characteristics of the simulation model are

1. Tasks have fixed worst case execution time (WCET)
2. Deadlines equal to their periods.
3. Tasks are independent.
4. All tasks are periodic.
5. System overheads will not be considered.

8.4 Scheduling Tasks in Real–Time Systems

There are many scheduling techniques that available out there and can be used in real–time systems. Some of them are widely implemented while some of them may be not suitable for modern real–time systems. Implementing a scheduling technique into real–time system depends on many factors and considerations. We will give you basic ideas about real–time scheduling in the following, covering common hard real-time scheduling techniques.

Scheduling techniques that widely used in real–time systems can be categorized into 2 main categories which are static priorities scheduling algorithm and dynamic priorities scheduling algorithm.

Figure 5 shows taxonomy of real-time scheduling.

![Diagram of scheduling techniques for real–time systems]

Figure 5. Scheduling techniques for real–time systems

Each category can also be divided into subcategory such as preemptive dynamic schedule technique that considers deadline and preemptive static schedule technique that considers response time, computation time or periodic time but with limited time space we can not put all of them here.

**Dynamic Priorities Scheduling Algorithm** Dynamic priorities scheduling algorithm has a simple goal, to maximize CPU usage. Moreover, it also has some advantages over static priorities scheduling such as changing priorities can be done on the fly by the scheduler.

Re–ordering a job happens based on many factors. 3 important criteria are
Earliest deadlines
- Computation time
- Laxity

**Earliest Deadline First (EDF) Scheduling Algorithm**

Earliest Deadline First Scheduling Algorithm is one of preemptive dynamic scheduling techniques that consider deadline. As the name implies, in this algorithm, the job with earliest deadline gets the highest priority. Priority of given tasks could be changed whenever users or applications want to. One of the advantages from EDF is that CPU utilization could reach 100 percent of schedule bound.

However, it also has some disadvantages. One among them is that there is no guarantee that which of the given tasks would fail due to overload of the system. This drawback is also one of main issues in this research.

![Figure 6. Example of Earliest Deadline First Algorithm](image)

**Schedulability Test for EDF**

Liu and Layland (1973) analyzed utilization based schedulability test in their paper and gave this equation for EDF in the case of $T_i$ equal to $C_i$.

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

As the equation has shown, this schedulability test is simple and easy to calculate. To test schedulability in our experiment, we also use this method.

**Static Priorities Scheduling Algorithm**

In this scheme, priorities of tasks are set at compile time. System designer or programmer is the one responsible for these timings. There is no possibility of change in priorities when the system is running. It is more restricted but somehow suitable for sensors and actuators when priorities are known prior to system initialization.

**Rate Monotonic (RM) Scheduling Algorithm**

Rate Monotonic Scheduling Algorithm is an example of priority-based preemptive static scheduling techniques. In this algorithm, we concern the rate or period of tasks, shortest period of task will be given the highest priority.

RM is also called static algorithm because it can not be changed once priorities are assigned to tasks. CPU Utilization of given tasks in this algorithm could not reach 100 percent because it is computed as a sum.
9 Simulation approach

Previous real-time approaches in Ethernet were proprietary, certain changes introduced by those vendors at the application level and switching hardware were not reproducible due to closed source or patented technologies, therefore the demand for a real-time Ethernet with open standards is increasing.

It is not possible that normal switches can co-exist in a real-time Ethernet network, therefore the replacement of existing switches is inevitable, to offset those costs, the only way is to introduce the least changes as possible to equipment that needs those modifications.

Ensuring those modifications are based on open standards at the very least is an important issue as it will ensure widespread adoption when compared to proprietary technologies.

To gather a better understanding into the workings of open technologies exclusively, simulation models were applied to find its feasibility in a real-time network environment.

Most of the technologies that are presented in this project approach are proven technologies, while not necessarily applied to real-time Ethernet before, all of them posses special features that are required to make real-time Ethernet a reality, in the simulations that is, for now.

9.1 Real-Time Ethernet Frame

Previous topic gives you a roughly idea about the Ethernet protocol and its frame contents, referring to a normal Ethernet frame in figure 3, we model by modifying an Ethernet 802.3 Frame by adding 5 more fields to it, while preserving standard 802.3 fields and length as we intend to make it compatible with 802.3. Modifying a field or its length, which is not preferable, causes changes in hardware and underlying software. Figure 7 shows a real-time Ethernet frame.

Similar approach in modifying 802.3 frame was introduced by Andres Kallerdahl in his paper[4].
As you can see, the size of each field is not mentioned in the simulations, doing so is not necessary and makes our simulation needlessly complex. We will define the frame size in its totality instead, the size of an Ethernet frame is important to the simulations because it must be used to calculate transmission time and to keep real time Ethernet frames compatible with Ethernet 802.3 frames, such a frame can not be larger than 1518 bytes. Because there is not need to assign of each field, it is an important assignment and must be used to calculate the transmission time of the frame. In the model, instead of assign to each field, we assign the size of frame.

### 9.2 Real-Time Layer

The real-time layer is inserted into the OSI model between application layer and the datalink layer because in our experiment the rest of the OSI model will not be considered. Figure 8 shows end-to-end communication with 2 nodes and 1 switch, real-time layers are inserted to frames at each node.

![Real-Time Layer Diagram](image)

**Figure 8. End-to-End Communication with Real-Time Layer inserted.**

### 9.3 Real-Time Switch

Real-Time Switch or RT Switch is an Ethernet switch with capability to serve and supervise both real-time traffic and non real-time at the same time, when compared with a normal Ethernet switch, a real-time Switch has the following extra features.
- Management Software is the software inside a Real-Time Switch that has 2 functionalities: test feasibility of packets and supervise real-time.

- Real-Time Layer which is the new layer inserted into the existing OSI model capable of sorting packets according to Earliest Deadline First algorithm.

### 9.4 Handling of Real-Time Traffic

These steps will explain how the Real-Time Switch will handle real-time traffic from nodes in the network as illustrated in the state diagram, figure 5.

1. A node generates real-time traffic.
2. Real-Time traffic travels along the connection to real-time switch.
3. Real-Time traffic reaches a real-time switch. The real-time switch calculates feasibility based on simple schedulability test proposed by Liu and Layland. If it is feasible, the real-time switch will forward the traffic to its destination.
4. Destination node calculate feasibility between the real-time switch and itself, regardless of feasibility, it will reply but with a distinctive value in the response field of feasibility frame.
5. Real-Time switch will again calculate feasibility of replied traffic. If it is feasible, the real-time switch will assign a "Channel ID" and subsequently forwarded back to its origin.
6. Origin node now begins generating real-time traffic utilizing the "Channel ID" assigned earlier by the switch.

![Figure 9. Real–Time Traffic Handling Diagram](image-url)
9.5 Deadline Partitioning

Deadline Partitioning Scheme was introduced by Hoai Hoang[4]. This scheme works by dividing the deadline of a job into 2 for decisions, made by the real-time switch. Figure 10 shows deadline partitioning scheme.

\[ T_d = T_{d1} + T_{d2} \]

The real-time switch makes decision based on deadline of jobs whether it should be forwarded to its destination. When the real-time switch receives a packet carrying a job. The real-time switch will divide deadline into Td1 and Td2. Then Td1 will be checked for its schedulability according to the equation give by Liu and Layland, then it will be checked if the time taken from origin to the real-time switch is greater than Td1. If 2 conditions are not satisfied. The real-time switch will not forwarded or secure the requested channel. The same principal also applies for Td2.

9.6 Switch Limitation

In case of high load or when the buffer is full, Ethernet switches have no option to treat incoming packets that can not be enqueued except dropping them, doing this is acceptable in normal situations where TCP is implemented to handle dropping and notify upper layers, but in real-time communication such action is unacceptable and definitely leads to damages.

To avoid dropping of real-time traffic, in a real-time switch, the predictive mechanism in management software to monitor buffer and output queue usage to assure that every established real-time channel must be enqueued and must be delivered to destination within delay bound.

10 Software Technology

10.1 Introduction

Since the aim of this master thesis is to develop real-time Ethernet simulation model within platform. It is the responsibility of the simulation
software to provide an environment comparable to real-time. This section of the paper will guide you through the details of software that will be used in the experiment.

10.2 Software Simulation and its benefits

Computer simulation seems to be a smart technique of choice to visualize experiments. It has been popular for decades especially in computer science and engineering because using computer simulation reduces developing time considerably. Moreover, there is no need to create a prototype which is a time-consuming process and cost of investment is quite high. Results attained from computer simulations are accurate and close to real results. However, accurateness depends on many factors but outside the scope of this project. With those mentioned benefits, computer simulation is the method of choice when one would like to get results from an experiment.

The simulation is especially suitable for our case since existing technologies and theories are applied to our simulation with few variables, moreover due to the real-time nature, the simulation can depend on the internal clock of the simulation application to mimic a real time clock, although the host might not be real-time, all procedures are anchored to the clock of the simulation application which will ensure strict timing across the board even if the host has processing delays, therefore testing for feasibility in our approach can be called reliable.

10.3 OMNet++

OMNet++ is a simulation framework for discrete events which can be used to simulate any model where the discrete events are applied.

In a discrete event simulation model, we consider events as they progress through time. For example, a packet flows from a client to another. In between, there is no interesting event happening except time delay except. Therefore, a discrete event simulation such as OMNet++ is suitable to simulate communication networks and queuing systems.

Processing resources at the host conducting simulation and the speed of simulation are major concerns when it comes to realizing the simulation model into a production real-time Ethernet switch since the timing and resources at hand in the simulation must exceed or match that of the switch for the simulation to be effective, although this is outside the scope of the simulation, these concerns have been taken into account.

To achieve the fastest simulation possible on a computing system, the compiler of the simulation program plays an important role. To this end, C and C++ are both scientific proof that they are the fastest human-friendly compiler available today. OMNet++ itself has been developed with C++, object-oriented programming features found in C++ also benefits users in
term of creating modules. To implement the different pieces of the simulation model, seeing and treating it as an object would be easier for user to interact its it and to communicate with the simulation framework.

10.4 INET Framework

INET Framework is a suite of communication modules consisting of many useful modules ranging from the physical layer to application layer of OSI model. It works as a supplement to OMNet++ enabling developers to construct new models with limited time. It as developed and distributed under GNU General Public License by Andras Varga, the same person who developed OMNet++. Today, It is hosted and maintained at Monash University, Australia.

The Ethernet module that has been used in our simulation model is just one of numerous modules that the INET Framework provides. It consists of several submodules to effectively mimic behavior of Ethernet and covers all main Ethernet functionalities.

These are submodules that come with Ethernet modules from INET Framework.

1. Ethernet Client Application : Works as traffic generator
2. Ether Server Application : Works as processing unit
3. Ethernet LLC Module : Provides Encapsulation and Decapsulation from higher and lower layer respectively.
4. Ethernet MAC Module : Responsible for Layer 2 routing.
5. Ethernet Switch Module : This module consists of Ethernet MAC Module Module and Relay Unit which is responsible for bridging and switching.

The Ethernet module provides a framework to our simulation. In order to achieve our goal, we have developed EDF onto the Ethernet module from INET Framework. The modification is performed compliant with GNU General Public License, which can be found in Appendix F.

10.5 Software Design

Due to the fact that both client and server have the same structures in terms of hardware and software. The term client-server is used to determine logical behavior of a node only. A client is considered as a service requester while a server is considered as a node which servers such service to the client.

Those characteristics of a typical client-server model allow us to simplify the flowchart of the simulation model by organizing the flowchart in block
diagram. A node, no matter what its function is, either a client, server or switch, will contain server building blocks. Details of building blocks of each node will be described in the section "Simulation Structure". The flowchart in Appendix C will represent who to simulation works and how each layer interacts with neighboring layers.

11 Experiment

11.1 Introduction

As mentioned earlier that if a task could not meet its deadline, severe damages are likely to occur and in order to prevent such things. Extensive tests must be performed to assure that 100 percent of the generated real–time packets from any source to any destination must be delivered before their deadline or simply speaking they must be feasible.

The aim of several experiments that we are going to perform is to show the performance of an Ethernet switch with EDF implemented and show how many percentage of request can be guaranteed.

To achieve such goal, software simulation has been chosen to prove and visualize our experiments.

11.2 Time

Simulation is the process of imitating real events as mentioned earlier, to attain output faster, means that the simulation advances in time at different rates. Thus, different meanings of time co-exists in this context.

The time in this simulation can be distinguish into 2 types:

- Real time is clock time.
- Simulation time, virtual time or model time are time to represent real time in the simulation.

When simulating a network to see throughput for example at hour 4th. It is said that 4th is the simulation time, model time or visual time. It does not happened in reality but it is forecast of what will occur based on input time as an assumption.

11.3 Calculation

11.3.1 Feasibility Test

Feasibility test performs at the real–time switch and at server side to assure that real–time traffic will be handled within delay bounds.

$$MaximumDelay = D_i + ChannelLatency$$
Where Maximum Delay is acceptable time taken from a client to real-time switch or real-time switch to a destination.

$Di = \text{Relative Deadline}$

Channel Latency = medium delay

Feasibility test is performed using this simple formula from Liu and Layland.

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Where N is number of connection.

Each packet will be tested against the formula, if the packet passes feasibility test from a real-time switch then it will be forwarded to destination. The destination again will check for feasibility of the link and all active connections to it. If it feasible, it will generates a response telling the origin to begin sending into the real-time channel. Figure 11 shows how a server handles more than 1 real-time channel. (Please see flowchart for the details)

![Sample Network Diagram](image)

**Figure 11. Sample Network Diagram**

### 11.3.2 Throughput

Throughput is the amount of work or task of the computer, or in this case the processing capabilities of the server at a given period of time. Throughput is calculated using this formula.

$$Throughput = \frac{FrameSize}{t \times 10^6}$$
Frame size is amount of frame including all headers which is fixed at 1000 bytes for all tests. \( t \) represents the timer which starts counting when first bit arrives and stops when the last bit of a frame arrives.

**subsubsectionSpeed** This shows how to calculate the sending speed in Ethernet networks without EDF implemented.

On a network with 100Mbps of aggregate bandwidth and an average packet length of 1000 bits plus overhead of the Ethernet frame which is 96 bit, the time it will take 1 bit to travel across this network is as follows:

\[
\text{Speed} = 1096 \times 10 \text{ns/bit} \quad \text{Speed} = 10.96 \text{microseconds}
\]

However, 10.96 microseconds is the time elapsed for transmission of one packet to one hop when switch processing time equal to 0 and processing time at server node equal to 0.

For a network with one Ethernet switch like in our simulation model. Estimated round-trip time from any client to any server will be:

\[
\text{Time taken} = 10.96 \text{microseconds} \times 4 = 43.84 \text{microseconds}
\]

When switch processing time and CPU processing time at the end node are concerned the equation could be rewritten to

\[
\text{Speed} = 10.96 \text{ms} + \text{switch exec. time} + \text{CPU exec. time} + \text{prop. delay}
\]

Calculations that we stated here are derived from the theoretical part and they can be used to estimate arrival time in Ethernet networks.

In the simulation model with non real–time and real–time traffic co-existing, there are some factors that affect calculation results which are queuing time in each node which depends on congestion of the network. Thus, results from calculation might differ from a raw vector file context.

### 11.4 Simulation Structure

The network we consider in the experiment consists of one Ethernet switch and several nodes. Messages or Packets go from a host to the destination passing the Ethernet switch. The Ethernet switch works with store-and-forward forwarding mechanism.

The simulation model consists of these main elements.

1. Topology Files: Topology files are text-based files describing how things are connected ranging from simple and compound modules inside of every node to connectivity among nodes in the network.

2. C++ Source code: C++ source files describe how each simple module works. They are the heart of the simulation model which controls all behaviors of the model including packet generation, and processing packets from higher layer, etc.
3. Modules and Submodules: Modules are smallest components in the model. They are programmed with C++ along with its library. Components can combined together to create a node in layer fashion that reflects an OS model and can be understood easily but some unnecessary layers are eliminated to suite our needs.

4. Simulation Parameters: All simulation parameters are defined in the main configuration file of OMNet++, omnetpp.ini. Those parameters can either be coded directly to source code so called hard-coded or defined in main configuration file. The author prefers the second way since it is more formal and parameters can be changed freely with any text editor.

11.5 Network Topology

The network consists of one Ethernet switch with several nodes as shown in figure 12. The figure 12. show a sample network diagram. However, in our experiment the number of nodes may differ from this diagram.

Figure 12. Sample Network Diagram

Node: There are 5 modules in each node which are client module and server module which represents the application in OSI model, real-time, LLC and MAC module respectively. Application layer module is responsible for traffic generation, client module at the application layer reads parameters from the OMNet++ global configuration file omnetpp.ini. For processing a request, the server module at the application layer process a request depending on its processing time, subsequently sending a reply back to the sender.

Figure 13 shows the modules inside a node while figure 14 shows modules of the switch.
**11.6 Logical Connections**

Logical connections describe how each node in the network behaves or more specific how each node sends and and receives packets and how they react to the packets.

Accurate results are pivotal to our simulation, so tests were designed with different logical connections within the network. Each test result has a figure to represent the logical connection of the network.
11.7 Network Traffic

Along with real-time traffic, non real-time traffic was also generated in our simulation. Co-existing traffic types would be beneficial towards producing accurate results.

To reduce developing time and complexity of the simulation model, a traffic generator is designed as a part of the application layer which reflects behavior of Ethernet nodes.

11.8 Simulation Environment

Even inter-networking devices in the market today are built based from the same standard which is OSI model but some of them also implement special features to make their products more attractive to consumers.

In order to make our simulation more understandable and self-descriptive to others for reproduction purpose, features and parameters of devices must be clear-cut. So, in the future if one would like to re-simulate either by OMNet++ or other network simulators, they should be able to do so easily and results should be identical.

All tests are performed with these parameters in mind, some tests might have varying parameters but they will be declared explicitly to that end.

- There is only one processing unit in the Real–Time Switch with shared memory architecture.
- There is one shared queue in the Real–Time Switch.
- The Real-Time Switch operates with full–duplex mode.
- Buffer is equal to 1 megabyte.
- Switch forwarding delay or processing time for each frame is fixed at 12 microsecond.
- Queue length of datalink level is 100.
- Queue length is independent to buffer size.
- PAUSE frame will be sent out to all ports of the real–time switch if buffer is overflowed.
- Switch operational mode is store–and–forward. Store–and–Forward works by reading entire Ethernet frame before start sending. OMNet++ supports only this kind of modeling and it is one of limitations of OMNet++
These parameters are global parameters which effect every test. There are some parameters that change test by test. Those kinds of parameter will be described along with each test.

There is an important thing that should be noted here. All graphs are based on functions of 2 distinct values on x-axis and y-axis but some may not look like x-y graph because the congestion of raw data, especially those graphs that are generated from OMNet++. To understand these graphs, Appendix E serves as a guideline and points out how to interpret the graphs.

11.9 Simulation Results

11.9.1 Test 1

In this test, there are 3 clients and 2 servers. The clients generate only real-time traffic to the network.

The objective of this test is to show how EDF works in Ethernet network. HostC, HostD and HostE play the client role while HostA and HostB play the server role. Logical connection diagram in Figure 15 shows sending and receiving nodes.

![Network Diagram](image)

Figure 15. Network Diagram
Figure 16. Show Logical Connections of The Network

HostC, HostD and HostE generate real-time traffic using these parameters:

<table>
<thead>
<tr>
<th>Route</th>
<th>Period</th>
<th>Execution Time</th>
<th>Priority</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostC–HostA</td>
<td>0.03</td>
<td>0.010</td>
<td>1</td>
<td>0.33</td>
</tr>
<tr>
<td>HostD–HostA</td>
<td>0.04</td>
<td>0.010</td>
<td>2</td>
<td>0.33</td>
</tr>
<tr>
<td>HostE–HostB</td>
<td>0.05</td>
<td>0.005</td>
<td>1</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table 1. Traffic Generation Parameters

Figure 17 shows that 3 nodes request to establish 3 real-time connections and all of them are accepted as shown by the graph. The gap is time between requesting and assigning channel ID at the switch.

Figure 17. Requested and Served Channel
Buffer usage occupies approximately 0.3 percent as shown in figure 18 which is quite low.

![Figure 18. Buffer Used](image)

Throughput at HostA and HostB are the same at 96 Megabit per second and because throughput is almost constant value during the simulation time, there is no need to plot the result.

11.9.2 Test 2

2 more nodes have been added to this test. There are now 7 nodes in total. 2 new nodes generate only non real-time traffic to the network. Figure 19 shows network diagram along with Figure 20 which shows logical diagram of the network.

![Figure 19. Network Diagram](image)
2 new nodes generates non real–time with a mean value 0.5 and standard deviation of 0.05 in HostE and mean value of 0.6 and standard deviation of 0.06 in HostG. Real–Time packet generation parameters remain the same as Table 2 has shown.

<table>
<thead>
<tr>
<th>Route</th>
<th>Period</th>
<th>Execution Time</th>
<th>Priority</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostC–HostA</td>
<td>0.03</td>
<td>0.010</td>
<td>1</td>
<td>0.33</td>
</tr>
<tr>
<td>HostD–HostA</td>
<td>0.04</td>
<td>0.010</td>
<td>2</td>
<td>0.33</td>
</tr>
<tr>
<td>HostE–HostB</td>
<td>0.05</td>
<td>0.005</td>
<td>1</td>
<td>0.10</td>
</tr>
<tr>
<td>HostF–HostA</td>
<td>0.3,0.03</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>HostF–HostB</td>
<td>0.5,0.05</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 2. Traffic Generation Parameters

In this test, we have both real–time and non real–time traffic in the network. Since real–time traffic will be treated first, throughput will be higher than non real–time traffic without doubt. Figure 21 shows the result from simulation. The graph shows clearly that real–time traffic throughput is much more higher than non real–time traffic.
Figure 21. Real–Time Traffic Throughput VS Non Real–Time Traffic Throughput in HostA

Real–Time switch was able to handle all 3 real–time channels. Buffer usage also remains that same.

11.9.3 Test 3

Figure 22 shows logical network diagram

Figure 22. Show Logical Connections of The Network

From the same network diagram and logical connection as Test 2, we lower the period of all real–time traffic and non real–time traffic.
### Table 3. Traffic Generation Parameters

As traffic increased, non real-time traffic has been forced to run at lower throughput by real-time traffic. Figure 23 and 24 show throughput of HostA and HostB.

![Figure 23](image)

**Figure 23. Real-Time Traffic Throughput VS Non Real-Time Traffic Throughput in HostA**
The Real–Time Switch could be able to server all requested channels without any problem as Figure 25 has shown.

Buffer usage is 0.3 percent which is the same amount as previous tests as Figure 26 has shown.
11.9.4 Test 4

Number of nodes in this test has increased to 12 nodes in total. Every node in this test generates only real-time traffic to the network. There are 3 servers waiting to process tasks from 9 clients as Figure 27 has shown.

Figure 27. Network Diagram

Table 4 shows packet generation parameters and how traffic flows.
<table>
<thead>
<tr>
<th>Route</th>
<th>Period</th>
<th>Execution Time</th>
<th>Priority</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostC–HostA</td>
<td>0.002</td>
<td>0.0005</td>
<td>1</td>
<td>0.10</td>
</tr>
<tr>
<td>HostD–HostA</td>
<td>0.004</td>
<td>0.001</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>HostE–HostA</td>
<td>0.008</td>
<td>0.004</td>
<td>3</td>
<td>0.50</td>
</tr>
<tr>
<td>HostF–HostB</td>
<td>0.0016</td>
<td>0.0004</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>HostG–HostB</td>
<td>0.004</td>
<td>0.0005</td>
<td>2</td>
<td>0.125</td>
</tr>
<tr>
<td>HostH–HostB</td>
<td>0.008</td>
<td>0.0032</td>
<td>3</td>
<td>0.40</td>
</tr>
<tr>
<td>HostI–HostL</td>
<td>0.003</td>
<td>0.0015</td>
<td>1</td>
<td>0.33</td>
</tr>
<tr>
<td>HostJ–HostL</td>
<td>0.004</td>
<td>0.001</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>HostK–HostL</td>
<td>0.005</td>
<td>0.001</td>
<td>3</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 4. Traffic Generation Parameters

Figure 28 shows that all 3 real-time channel have been accepted.

![Graph showing served and requested real-time channel](image)

Figure 28. Served and Requested Real-Time Channel

In this test, the switch is still able to serve all requests as expected but there is one thing that should be noted. Buffer of the real-time switch has doubled when compared with previous test from approximately 30 percent to 60 percent. Figure 33 shows buffer usage in this test.
Please note that result from figure 29 is plotted by taking an average value instead of plotting every record from the raw vector file. This is because the system could not be able to complete the request due to limitation of the system.

In this test, throughput of all hosts is approximately 96Mbps.

11.9.5 Test 5

Number of channels has been increased to 20 in this test. There are no figure of network diagram and logical diagram because it is space-consuming to put those figures and Table 5 can be substituted.
Table 5. Traffic Generation Parameters

Table 5 shows that there are 20 real-time communication channels in total. Each link handles 5 real-time channels. Every channel has the same parameter with execution time equal to 0.004 or 4 milliseconds and its period equal to 0.040 or 40 milliseconds.

Figure 30 shows that all 20 requested channels are accepted. The last channel is established at approximately 0.4020. Even though all are accepted, however when comparing this with previous test, the time gap between a requested channel and served channel is wider. It likely caused by traffic congestion as more nodes were added.
Figure 30. Requested channels vs Accepted channels

Buffer usage peaked at 2000 bytes as shown in figure 31. When compared with previous test it is clear that in this test buffer usage is lower. It is likely that the high buffer usage is caused by traffic congestion.

Throughput for all 4 servers is 96Mbps.

![Buffer Usage](image1)

Figure 31. Buffer Usage

11.9.6 Test 6

From all previous tests, the percentage of real–time channel acceptance is 100 percent. No matter if the network handles only real–time traffic, non real–time traffic or both.

In this test, there is only one server that processes real–time tasks from 40 clients or nodes. Each client generates only one task. In total, there will be 40 tasks on the physical link from switch to server.

Real–Time tasks have 80 milliseconds as their periods and 1.5 milliseconds as their execution time. Utilization from physical link will be 0.75 percent.

Figure 32 shows that all of requested channels are accepted.

![Channel Acceptance](image2)
Figure 32. Real–Time Channel

Buffer usage is a little bit higher than Test 5 as shown in figure 33.

Figure 33. Buffer Usage

Results clearly show that all real–time channels are accepted. the Ethernet Switch is able to handle real–time traffic very well.
At client nodes, tasks occupy approximately 1 percent of utilization.
A Utilization at that port jumped to approximately 3.3 percent.

At the Ethernet Switch, tasks occupy approximately 1 percent except one port which handles 40 real-time channels from all 40 clients to Server A. Utilization at that port jumped to approximately 3.3 percent.

Figure 34. Datalink Utilization at clients.
In this test, we double the number of clients from 40 to 80. There are 2 servers each of them processes 40 tasks from 40 clients. Each client has its period equal to 80 milliseconds and execution time equal to 1.5 milliseconds. Thus, each physical link to server A and server B has a utilization of 0.75 percent. This test has been setup to analyze how the Ethernet Switch performs when real-time traffic increases.

Figure 36 shows that all requested channels are accepted.
For buffer usage, figure 37 shows buffer usage in this test which is the highest level since Test 1. It can say that buffer usage depends on the number of clients or more precisely traffic on the network.

![Figure 37. Buffer Usage](image)

11.9.8 Test 8

Up to now, there has been no packet drop and there is no parameter that seems to make huge impact to real-time switch.

In this test, we decided to reduce number of queue length from 100 to 20 and inject traffic from 40 nodes to one server.

![Figure 38. Requested and Accepted Channels](image)

Figure 38 shows that the number of accepted channels dropped to 20 because management software inside real-time switch detect queue length
and does not allow the number of accepted exceed queue length to minimize the risk of packets being dropped due to a full queue.

There is no packet drop in this test. Thus, there is no reason to plot linear graph starting at zero.

Buffer usage and utilization of port in graphical format also weaved since the number of nodes is lower than previously tested. It is not possible that those value will go higher.

11.9.9 Test 9

In order to test how the queue length effects real–time traffic, we have modified management software inside real–time switch not to detect queue length.

![Requested and Accepted Channels](image)

**Figure 39. Requested and Accepted Channels**

Figure 39 shows that accepted channels account for only 25 channels while number of requested channel are getting higher because unestablished channels were trying to ask for acceptance.
Figure 40 shows the number of frames dropped at MAC layer on each port of real-time switch. As more traffic congest prevail, more drops occur at MAC layer.

Future Development

1. UDP and TCP traffic patterns: Current version of the simulation model supports the OSI physical layer to network layer. Adding TCP and UDP traffic generation functions could be useful and bring simulation results closer to real-world network behavior.

2. Automated network topology generation: Complexity of network topology could be generated automatically.

3. XML extension: Even XML is already supported by OMNeT++, but it was not used in this project.

4. Real-time graphical display: The simulation results of simulation have to be analyzed only after simulation is done. Real-time graphical display is not used in this project.

The results from the simulation model are accurate and satisfying but in a short period of developing time, we could not add some additional features to achieve a perfect simulation.

As mentioned earlier, this simulation is modular and extensible. Anyone interested in real-time simulation with OMNeT++ and basic background knowledge in C++ should be able to extend features and functionalities who interested in real-time simulation with OMNeT++ and basic background knowledge in C++ can extend the simulation model and add new features to improve the simulation results. The results from the simulation model are accurate and satisfying but in a short period of developing time, we could not add some additional features to achieve a perfect simulation.

11.10 Future Development

The results from the simulation model are accurate and satisfying but in a short period of developing time, we could not add some additional features to achieve a perfect simulation.
5. Using different buffer size and processing unit model could generate more accurate results and more specific towards some brands of Ethernet switches in the market.

6. Different queuing model such as separated queue for each port of the switch.

7. The simulation model was conducted with simulated performance and network traffic without concerning CPU congestion from any other aspects such as CPU load from local applications. To get a more accurate result, one should include those situations into the simulation model where those could apply.

12 Problems Encountered

During development process, many problems occurred and those effected directly to developing time.

- Available Resources: Real–time communication is a general topic that is widely available else where in any form of media but for finding information specific topic like implementing Earliest Deadline First algorithm into Ethernet or supporting deterministic in Ethernet is not an easy task.

- Environment parameters: Parameters used in experiments are assumed based on books, PDFs and papers that the author can find because the author does not have real experience in industrial automation.

- Debugging: OMNet++ does not output identical debugging messages on every platform or development environment. For example, debugging functions perform well on Unix variants but on Windows platform with Cygwin it does not produce any messages when simulation failed to run. Thus, there is no way to identify the problem until the same procedure is re-run on Unix variants.

- Compiling: To test portability of codes, codes have been compiled and run on several computers with different platforms. Such process takes time because preparing an environment for OMNet++ to run differs on each platform.

- Processing Power: With processing power of Intel 1.4GHz on a Laptop platform, it can say that is not powerful enough to simulate a network with many nodes. Result vector file is another issue because OMNet++ produces ten of thousands records on a vector file. To plot a graph from that file, it takes time and sometimes cause plotting software to stop running.
13 Conclusion

In this master thesis project, the basic requirements for a real-time Ethernet network has been examined and tested with several simulation models. In one of the simulation cases the clients function as a sensor or actuator generating periodic tasks to the server. The most issue will arise at the Ethernet switch or server, where the traffic is most congested.

According to the simulations present new insights into the traffic characteristics. Therefore new responsibilities that the switch and server need to handle in real-time Ethernet network should be considered especially the number of real-time channels the switch can handle at any one moment should be announced to surrounding nodes which is based on the buffer size and queue length of datalink layer within the switch, effectively refusing to open more real-time channels than the switch could handle.

The schedulability test suggested by Liu and Layland is not enough to prove that tasks will be always be feasible on real-time Ethernet. The system can be feasible only if all established channels are under supervision and controlled by special management software inside the Ethernet switch that supervises time constraints across all ports, it should also control whether the Ethernet switch should accept new real-time channels in different traffic environments.

The feasibility of real-time Ethernet depends on more factors than mentioned in this paper. Therefore extensive tests with varying parameters in live environments is recommended. When the Ethernet network becomes multi hop, synchronization using the precision timing (IEEE 1588) standard should be considered to augment packets with timing information so that the timing constraints remain predictable within acceptable margin.

References


14 Appendix

14.1 Appendix A Glossary

Real-Time

- Real–Time Switch is Ethernet Switch that capable of handle both real–time traffic and non real–time traffic.

- Real–Time Channel is end–to–end virtual connection between 2 nodes under supervision of Real–Time Switch.

- Real–Time Connection ID is unique number that indicates origin of real–time traffic.

- Real–Time Frame is Ethernet frame with several fields added.

- Real–Time Layer is special layer added to OSI model for special purpose. In the case to achieve real-time behavior.
- Real–Time Node is a PC or embedded system that capable of generating real–time traffic.

**Network**

- Propagation delay is amount of time Ethernet frame is delayed when it travels through connection or media.
- Date rate is specified number of bits that can be sent in a second it also can be used to calculate transmission time.

### 14.2 Appendix B Hardware Components

- Intel Processor running at 1400MHz
- 786MB RAM
- 30GB Hard drive
- Standard AGP Display Card
- 100 Mbps NIC
- Laptop form factor

### 14.3 Appendix C Software Components

The simulation model is made using OMNET++, the best well-known open source discrete event simulation environment, on Cygwin 1.5.21 for Windows to simulate Linux developing environment along with these required software components.

- Tcl/Tk 8.5
- ImageMagick
- GraphViz
- GifTrans
- C++ 3.4.4 (GCC)

### 14.4 Appendix D Flowchart

Flowchart explains only core functions of real-time properties. Properties of Ethernet such as medium access control and bridging are not covered here.
Figure D-1. Flowchart of application layer (Client)
Figure D-2. Flowchart of application layer (Server)
Figure D-3. Flowchart of Real-Time Layer
Figure D-4. Flowchart of LLC layer
Figure D-5. Flowchart of MAC layer
Flowchart: Ethernet Switch

Start

Ethernet frame arrives

Yes

Real-Time Traffic?

No

Send message to the sender telling that RT channel can not be established

No

Check if switch is capable to accept new connection

Put in the FIFO queue

Yes

Send a message to destination asking if new connect could be established

Wait for response

Receive message with answer

New RT Channel could be established?

Assign new RT Channel ID and tell sender to start sending packets

End

Figure D-6. Flowchart of Ethernet Switch
14.5 Appendix E Reading Graphs

Actually, graphs that appear in Simulation Result generated from OMNet++ are line graphs but because of too many records from raw data, they might look like bar graphs.

Take buffer level from Test 1 as an example.

![Figure E-1. Buffer Used](image)

As one can see that it looks very like bar graph. Now look at how many records from raw input as figure D-2 shows.
There are 32324 from input vector file. And now we take a look at some lines excepted from input file as shown in figure D-3.
First column represents simulation time while the second represents buffer usage. Now let take some of raw data for example 20 records and plot graph. Figure D-4 shows how it will look like.
Figure E-4. Sample Data

Base from graph which is plotted from only 24 records shown in figure D-4. When graph get congested, readers can imagine easily how graph will look like if we have around 32,000 lines or more.

14.6 Appendix F GNU General Public License Compliance

In order to comply with GNU General Public License, codes have been modified by according to these rules.

1. Person who changed the copies and date of change are mentioned.

2. Modified copies are available to public with free of charge.

3. Modified copies are available in readable form without executable form.

4. Since the simulation model does not need human interactive in command line mode. There will be no announcement printed out.

5. Anyone who is interested in this simulation model can download or extend functionalities on your own as long as you accept GNU General Public License.

Full details of GNU General Public License please visit http://www.gnu.org/copyleft/gpl.html
OMNet++ source code including C++, NED topology file, and message file are portable. Source code of all experiments have been compiled and run on following platforms.

1. AMD Sempron 1.6GHz x86 architecture with SUSE Linux 9.2 i386
2. AMD Opteron 2.0GHz x86 64 Bit architecture with SUSE Linux 10.0 for x86
3. Intel Celeron 1.4GHz x86 architecture with WindowsXP