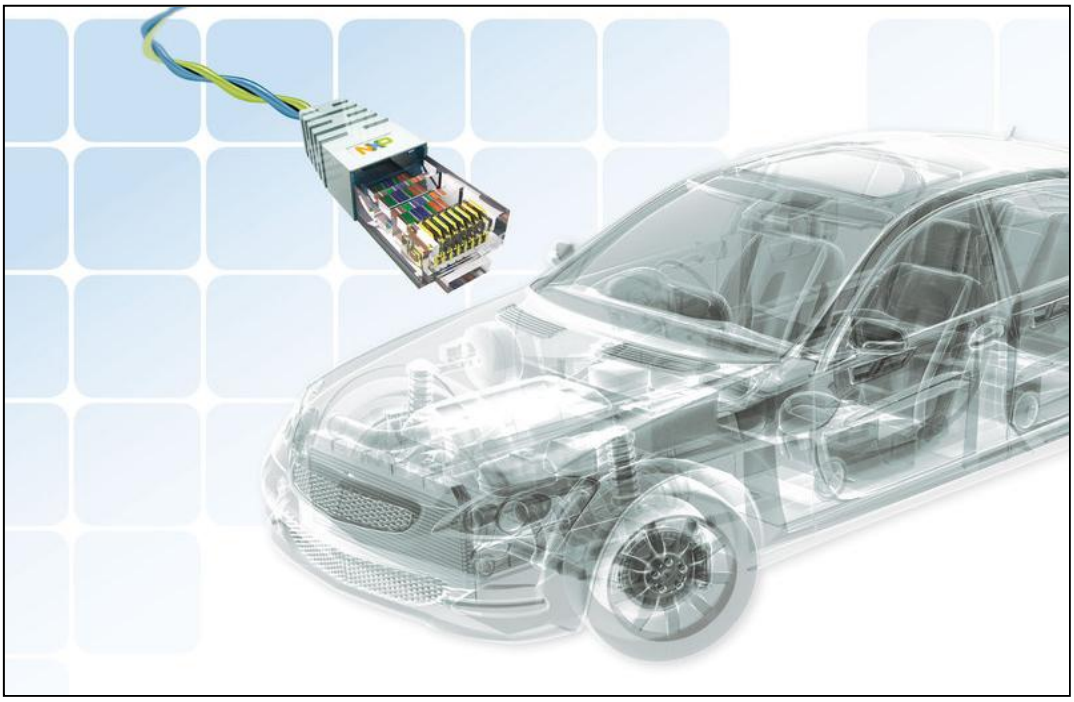
CHALMERS



Evaluation of switched Ethernet as a backbone for in-vehicle communication

Master of Science Thesis in Computer Systems and Networks

KOSTAS BERETIS
IEROKLIS SYMEONIDIS

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, June 2013

Evaluation of switched Ethernet as a backbone for in-vehicle communication

KOSTAS BERETIS
IEROKLIS SYMEONIDIS

Examiner: JOHAN KARLSSON

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
The cover illustration is a press image provided by NXP at a press release on November 09, 2011. Available at: *http://www.nxp.com/news/press-releases/2011/11/nxp-develops-automotive-ethernet-transceivers-for-in-vehicle-networks.html*

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

Acknowledgements

We would like to express our appreciation to Samuel Sigfridsson and Jerker Fors from Volvo Car Corporation for their insightful advice. Also, we would like to express our gratitude to Michael Svenstam from ArcCore for his support. Our grateful thanks are also extended to Niklas Angebrand, Tobias Johansson and Mårten Hildell for their help during the development of the software for the experimental boards. Furthermore, we would like to express our deep gratitude to Professor Johan Karlsson and Postdoc researcher Risat Pathan from Chalmers University of Technology for their helpful guidance and comments. Finally, we would like to thank all the people both at Volvo Car Corporation, as well as ArcCore for making us feel welcome.

Abstract

Ethernet was not originally designed for real-time communication. However, the growing need for available bandwidth combined with the fact that it is an open standard has made the automotive industry to consider it as a possible solution for in-vehicle communication. To enable Ethernet to be used in real-time communication, certain guarantees (i.e. end-to-end delay bounds) have to be provided. The purpose of this thesis is to propose a network architecture for in-vehicle communication and to develop a technique for analysis of end-to-end delay. In order to meet this objective, both theoretical and experimental methods were deployed. A theoretical model of the system under consideration was produced, based on network calculus. An experimental setup, which accurately reflected an automotive communication network, was implemented in order to evaluate the theoretical model. Three use cases were investigated for two different maximum transmission units. The results obtained by the experiments provided valuable feedback that redefined the analytical model. Although the end-to-end delay bounds proved to be higher than originally anticipated, the validity of the theoretical model was confirmed for the great majority of the experiments. As a conclusion, it was shown that Ethernet can be used as a backbone in a domain-oriented architecture as long as support for IEEE 802.1Q and 802.1p is provided both in hardware and software. Given the fact that future versions of AUTOSAR will support these standards and that implementation of such a system can be done with commercial of-the-shelf equipment, great potential is being presented for the designers of future vehicular communication networks.

Table of Contents

Acronyms

ABS	Antilock Braking System
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software Component
CAN	Controller Area Network
CAPL	CAN Access Programming Language
CFI	Canonical Format Indicator
CoS	Class of Service
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSMA/CR	Carrier Sense Multiple Access with Collision Resolution
ECU	Electronic Control Unit
EMC	Electromagnetic Compatibility
ESP	Electronic Stability Program
FCS	Frame Check Sequence
HMI	Human Machine Interaction
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LIN	Local Interconnect Network
MOST	Media Oriented System Transport
MTU	Maximum Transmission Unit
OEM	Original Equipment Manufacturer
QoS	Quality of Service
RTE	Runtime Environment
SWC	Software Component
TCI	Tag Control Information
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TPID	Tag Protocol Identifier
TTCAN	Time-Triggered Controller Area Network
TTP/C	Time Triggered Protocol with Sae Class C Fault-Tolerant Requirements
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network

1. Introduction

In-vehicle communication networks are evolving constantly. Due to rising requirements in terms of performance, safety and comfort an increasing number of electronic control units (ECU) are integrated within modern cars. The demand for faster, more robust and more dependable networks, results not only in increased complexity, but also in higher production cost.

The electronic control units are connected by application specific communication buses. The most traditional network technology used by the automotive industry is CAN. CAN has been used since the late 1980s and is a cost efficient solution with bandwidth up to 1Mbit/s [1]. The physical medium consists of two twisted unshielded wires and access to the shared medium is priority based (strict). A more recent communication bus is FlexRay which has higher bandwidth (10Mbit/s). FlexRay was developed for higher reliability in real-time communications.

The increasing need for bandwidth has led the automotive industry to consider Ethernet as a possible alternative to the existing network communication protocols. Indeed the automotive industry has introduced Ethernet in vehicles for diagnosis/updates and entertainment [2]. Ethernet is being further examined as a network technology for in-vehicle communication since it offers high bandwidth (up to 1000Mbit/s) while being cost-efficient. Furthermore, using IP and Ethernet different traffic types originating from different communication protocols can be integrated into a common backbone.

Ethernet was not originally designed for real time communications and does not offer any timeliness guarantees. However, Ethernet-based full duplex data transmissions could provide high bandwidth/high rate communication with relatively low jitter and high determinism [3]. Additionally various versions of Ethernet are widespread and have become successful in other industrial domains such as avionics and industrial automation [4]. The fact that Ethernet standards are non-proprietary, may lead to reduction of cost. The only way to guarantee bounds for end-to-end delay over Ethernet is to carefully analyze the worst case behavior of the network traffic. A static, a priori known network, such as an in-vehicle network is ideal for such analysis. To the best of our knowledge, attempts to explore these issues in the automotive context have been previously performed only by simulation techniques [5]. However, the behavior of such complex systems can only be realistically examined by an actual hardware experimental setup. In this way useful results about whether Ethernet can be used as the backbone network of an automotive system architecture can be obtained.

The research carried out in the framework of this thesis aims to evaluate the use of switched Ethernet as the backbone network of an automotive system architecture. The main objective of the thesis is to evaluate the end-to-end delay of the network and provide deterministic upper bounds. This was done by examining the systems behavior under six different use cases, provided by Volvo Car Corporation. Another objective of this thesis is to propose a system architecture for in-vehicle communication. Yet another objective is to produce a theoretical model of the system under examination. Furthermore, it is of great importance to also set up an experimental model, which accurately reflects the system architecture. In order to demonstrate that existing technology enables the usage of switched Ethernet in the automotive domain, commercial off the shelf switches will be used.

The scope of this thesis is to explore and fully understand network communications on a switched Ethernet network. Only the case of standard Ethernet will be examined. Furthermore any switch used will meet the IEEE standards. Ethernet will be used as a backbone for the communication between the domain masters, according to the proposed domain-oriented architecture. There are also certain limitations for this thesis. First of all, intra-domain communication over Ethernet will not be addressed. The redundancy and fault tolerance of the topology are also out of scope. Finally only the case of Fast Ethernet (100BASE-TX) Ethernet will be considered since Gigabit Ethernet (1000BASE-T) is not yet available for automotive use due to EMC issues.

The remainder of this thesis is organized as follows. The literature review regarding in-vehicle communication networks as well as network calculus is summarized within the second chapter. First, a theoretical review of the concepts combined in our work is made, followed the implementation specific choices made under the context of our study. Network Calculus and its basic concepts are also presented.

The methodology used within this thesis, is presented in the third chapter, giving information regarding the theoretical analysis and the experimental procedure. One of the main parts of the research, the modeling of the system, is then introduced. Basic information regarding the equipment used is given, followed by the experimental implementation details.

A thorough presentation of the results obtained by the six different use cases is made in chapter four. Theoretical results are compared with experimental results and findings of the project as well as key observations are then thoroughly discussed.

The thesis is finalized with a chapter that summarizes the basic conclusions that have come up by the data analysis, trying to give answers to the questions presented in the thesis' objectives. The conclusions, apart from answering the basic research questions, result in the proposal of more researches in the future.

2. Literature Review

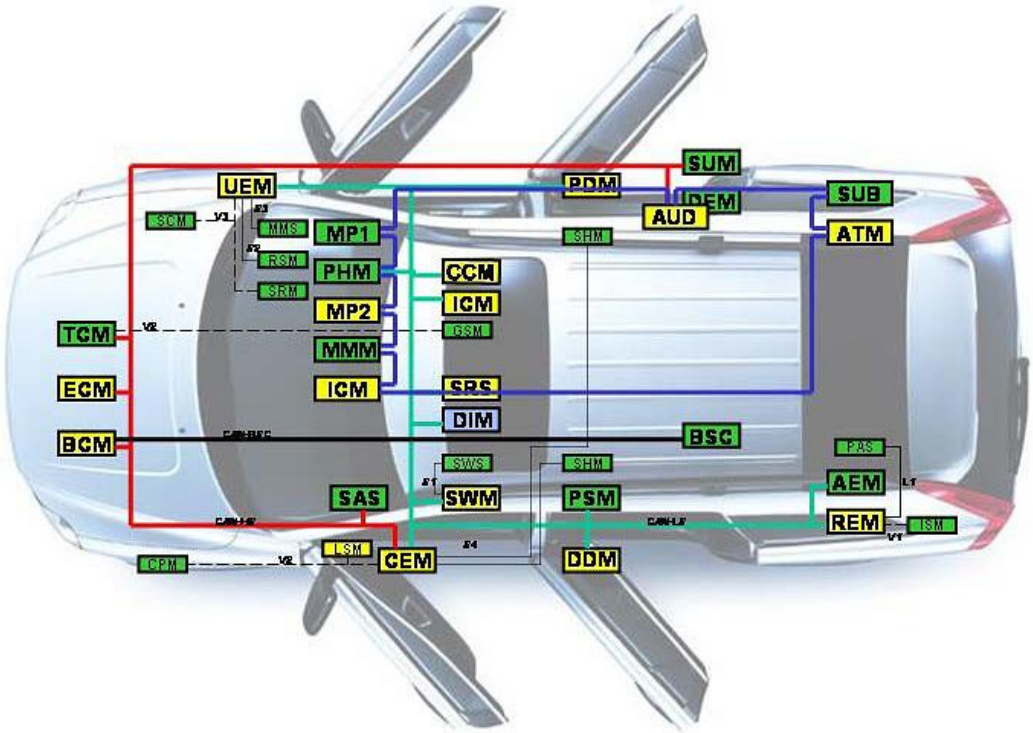
Over the past decades the number and complexity of electronic systems integrated in vehicles has increased exponentially. This was the result of replacing mechanical and hydraulic parts with ECUs, sensors and actuators. By using embedded systems it was also possible to add functionality especially in areas such as dynamic driving control resulting in more reliable and robust control systems. Leen and Hefferman presented these trends of the automotive industry in [6]. In the same article it was stated that "the cost of electronics in luxury vehicles can amount to more than 23 percent of the total manufacturing cost". Thus the design of such systems is becoming more demanding.

2.1 Automotive domains

In order to deal with the complexity the domain-oriented approach was recently introduced, where subsystems of similar functionality are grouped into the same domain. The most common functional domains found in a modern car are:

- **Powertrain domain:** Control of the engine (generation of power) and transmission of power from the engine to the driving axis through the gearbox. Such systems have strict timing requirements and demand high dependability in communication.
- **Chassis domain:** Responsible for control of car stability and dynamics. Typical subsystems of the domain include suspension, braking and steering (ABS, ESP). Again the requirements for dependability and predictability are high.
- **Body domain:** This domain does not deal with functionality directly connected with driving. It includes systems that concern the comfort of the driver and passengers. Typical systems include climate control, locks, windows, seats etc.
- **Safety domain:** Control of systems like airbags, safety belts and impact sensors.
- **Telematics/Infotainment domain:** Includes services like in-vehicle navigation systems and multimedia entertainment systems (CD/DVD players, monitors). It also includes human machine interaction systems (HMI) such as dashboard and head-up displays concerning route and traffic information.

An example of a domain-oriented architecture is shown in Figure 1. The system consists of around 40 ECUs, which are divided into the powertrain and chassis domain, the body electronics domain and the infotainment domain [7].



Block	Powertrain and chassis	Block	Body electronics
TCM	Transmission control module	CEM	Central electronic module
ECM	Engine control module	SWM	Steering wheel module
BCM	Brake control module	DDM	Driver door module
BSC	Body sensor cluster	REM	Rear electronic module
SAS	Steering angle sensor	PDM	Passenger door module
SUM	Suspension module	CCM	Climate control module
DEM	Differential electronic module	ICM	Infotainment control module
Block	**Infotainment**	UEM	Upper electronic module
AUD	Audio module	DIM	Driver information module
MP1	Media player 1	AEM	Auxiliary electronic module
MP2	Media player 2	SRS	Supplementary restraint system
PHM	Phone module	PSM	Passenger seat module
MMM	Multimedia module		
SUB	Subwoofer		
ATM	Antenna tuner module		
ICM	Infotainment control module		

Figure 1. Volvo XC90 Network Topology and ECU explanation [7]

2.2 In-vehicle communication networks

Functional domains of modern vehicles have diverse communication requirements. That fact led the automotive industry to develop distinct communication technologies to satisfy the specific requirements. As the number of ECUs inside a car increased, so did the amount of wiring needed to interconnect them. As an example, it has to be mentioned that a luxurious car today may contain more than 4km of wiring, while a car manufactured in 1955 had only 45m [6]. Extra wiring results in added weight and increased fuel consumption. The development of serial based communication networks reduced the need of using discrete

wiring. Application-specific communication buses are integrated in the vehicle. The most common networks will be described in this section.

Controller Area Network (CAN)

CAN was originally developed in the mid-1980s by Bosch and is the most widely used automotive control network up to date. CAN uses carrier sense multiple access with collision resolution (CSMA/CR) protocol in order to control the access to the bus. This method resolves collisions in a deterministic way. CAN employs small sized (payload of 0-8 bytes) event triggered messages which are transmitted in a common bus. The data rate of the bus may vary from 10kbps up to 1Mbps.

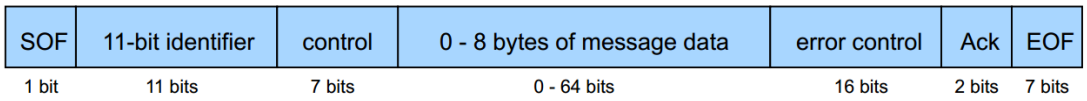


Figure 2. CAN message frame format

As it was mentioned CAN is a collision-based communication protocol. The sender transmits a message header with the 11-bit identifier. A collision-detect mechanism determines which node will be permitted to transmit a complete message. By assigning a proper identifier a deterministic bound on the queuing delay can be achieved. The collision resolution is performed by the "wired-AND". In case of simultaneous transmission by multiple nodes, if all nodes transmit a "1", then all nodes will see a "1". If at least one node transmits a "0", then all nodes will see a "0". When a node transmitting a recessive bit ("1"), detects a dominant bit ("0") it stops transmitting. The node that transmits all 11 bits of the identifier without detecting any inconsistency is the one with the highest priority and therefore can transmit the whole message.

Local Interconnect Network (LIN)

LIN is a time triggered, low cost serial bus which is typically found in systems within the body and comfort domain. Such applications may be car seats, door locks, sunroofs and in general on-off devices. LIN has a data rate of 20kbps and employs a master-slave mechanism. It is widely used because of its simplicity, low cost and the fact that it is an open standard.

Media Oriented Systems Transport (MOST)

MOST is the de-facto standard for multimedia and infotainment applications. Offering a high bandwidth of 25 up to 150Mbps, MOST uses a ring topology and synchronous data communication. Point-to-point links are used for data transfer and both synchronous and asynchronous traffic is supported. Synchronization between the nodes is

achieved through a master/slave mechanism, with one node being the timing master. The protocol was developed by a consortium of more than 50 companies including Audi, BMW and Daimler under the MOST Cooperation [8].

FlexRay

FlexRay is a fault-tolerant and highly reliable communication protocol. It combines both event and time triggered communication and offers data rates up to 10Mbps. A FlexRay network may consist of maximum 64 nodes. FlexRay supports dual channels and can be implemented in a star, bus or even mixed topology. The second channel can be used to provide redundancy or higher bandwidth. A FlexRay frame consists of a 5 bytes header and 0-254 bytes of payload. These frames can be scheduled for transmission during the static or the dynamic segment of a communication cycle.

The static segment contains slots which are statically assigned to nodes and offer guaranteed service. Multiple slots can be assigned to one node in the same cycle, which increases flexibility. The dynamic segment is divided into mini-slots. These mini-slots are assigned dynamically based on message identifiers and slot counters. A high message-id offers a guaranteed service and a low message-id offers a best effort service. In this way FlexRay combines both exclusive access to the medium, as well as arbitrary.

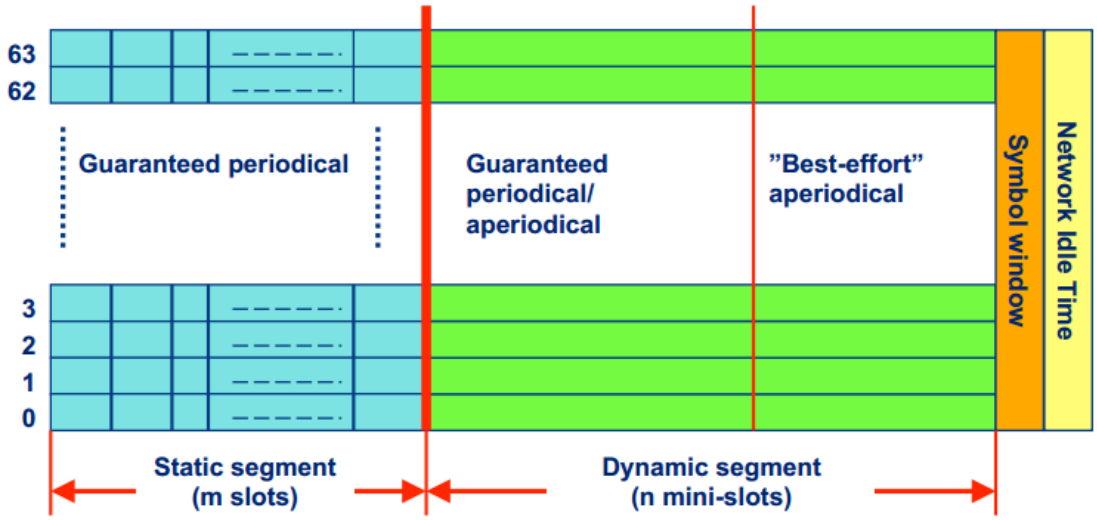


Figure 3. FlexRay Communication Cycle

Time-Triggered Protocol (TTP/C)

The TTP/C is a communication protocol developed by a consortium of companies known as TTA-Group. It offers speeds up to 25Mbps and employs Time Division Multiple Access (TDMA). For redundancy it deploys two channels and can be implemented either in

twin bus or twin star topology. Communication in TTP/C is organized in TDMA rounds. Each round is divided into slots and each slot is assigned to a respective node. The slots do not have to be necessarily equal within the same round, but they are statically assigned and do not change in each TDMA round. This static scheduling reduces the flexibility of the system but increases the predictability. Any non-periodic messages have to be fitted into the static slots by the application. Each TTP/C frame contains up to 240 bytes of data and 4 bytes of overhead.

TTP/C implements several mechanisms that make it a highly fault-tolerant communication protocol. An important feature is that clock synchronization is established through a global time base, without relying on a single time server. This makes the system fault-tolerant, which is required in safety critical applications such as avionics and x-by-wire systems.

Time-Triggered CAN (TTCAN)

CAN was extended with an additional layer in order to provide time-triggered behavior. TTCAN uses the same physical and data link layer as CAN and allows scheduling of messages in both a time-triggered and event-triggered way. TTCAN is implemented in a bus topology and achieves a data rate up to 1Mbps. Synchronization is achieved by a master node that sends periodic reference messages. Communication in TTCAN is achieved by a static schedule that consists of communication cycles. During each of the basic cycles TTCAN offers guaranteed service to periodic messages, while arbitration windows are used to support event-triggered messages. During these windows access to the medium is resolved arbitrary, as in CAN.

TTCAN employs the fault-tolerant mechanisms of CAN and also does not allow any retransmission of erroneous messages. Its main advantage is that it allows both event and time-triggered traffic on the same network, resulting in more flexibility. It is also a cost efficient solution because it was built on top of the CAN standard. However it has relatively low bandwidth which may limit its usage in the future.

2.3 Switched Ethernet

Ethernet was introduced in the 1980's as a networking technology for local area networks (LAN). It originally used coaxial cable as a shared medium which was later replaced by copper twisted pair and optic fiber cables. It quickly became popular and is the

most common LAN technology used today in data communication networks. Ethernet has a wide range of available bandwidth. Starting from 10BASE-T at 10MBps up to 10GBASE-T at 10Gbps. The most common type of Ethernet used today is the 100BASE-TX which offers a speed of 100Mbps. Ethernet is expected to exceed the speed of 100Gbps over optic fiber in the future [9].

Access to the medium is achieved through the carrier sense multiple access with collision detection (CSMA/CD) mechanism. Before transmitting a node senses the medium in order to see if another node is transmitting at that point of time. The node starts transmitting only when the channel is idle. During a transmission the node checks if a collision has occurred. If a collision is detected the node aborts transmission and sends a jamming signal informing all nodes that a collision occurred. The transmitting node then randomly selects a backoff period of time from a specified range of values, before attempting to retransmit. In case a second collision occurs the range of backoff values is increased exponentially and the same process is repeated. Therefore, the queuing delay of a message cannot be bounded. This fact made Ethernet unsuitable for real-time communications.

The introduction of switches allowed the segmentation of networks into collision domains. A collision domain is a segment of the network where collisions may occur. On one collision domain only one device may transmit at a time. Complete elimination of collisions was made possible with the introduction of full-duplex links which allowed devices to receive and transmit at the same time. In other words, a switched Ethernet network operating in full-duplex is a collision free network.

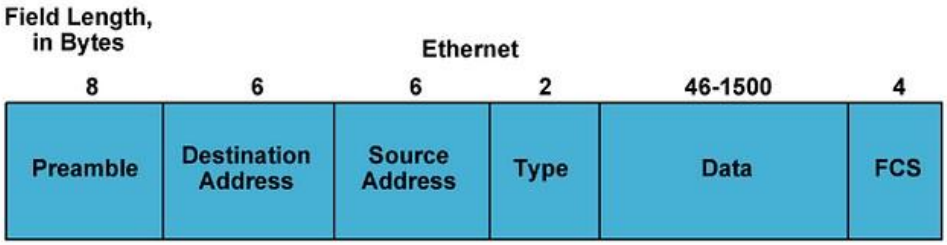


Figure 4. The Ethernet Frame

Before being transmitted, messages are encapsulated in the Ethernet frame. Figure 4 shows the structure of the Ethernet frame. A preamble of 8 bytes is used for synchronization purposes. The preamble is followed by the destination and source addresses, each of them having a length of 6 bytes. The following two bytes describe the type of the payload data. The payload may vary from 46 to 1500 bytes. Finally the frame check sequence field is responsible for error-detection.

IEEE 802.1Q added support for Virtual LANs (VLAN). By using VLANs a physical network can be segmented into multiple logical networks. This logical division is usually done by the switch. Each VLAN represents a broadcast domain. In a broadcast domain all nodes can reach each other using broadcasts at the link layer. A member of a specific VLAN cannot directly communicate with a member of another VLAN unless it is permitted by a router, which is a layer three device.

The VLAN tag as shown in Figure 5 contains all the necessary information for the switch, in order for it to make the correct forwarding decisions. The VLAN tag contains a 2 bytes field called Tag Protocol ID that is always set to the value 0x8100, in order to identify the frame as a tagged one. The following 2 bytes, named Tag Control Information, contains the User Priority field, the Canonical Formal Indicator and the VLAN ID. The User Priority field, also referred to as Priority Code Point (3 bits), allows up to eight priority levels, as defined by the IEEE 802.1p standard. In practice only four priority levels are supported by most switch manufacturers. The CFI (1 bit) is always set to 0, indicating that the MAC address is in canonical format. This was used for compatibility in token ring networks. Finally the VLAN ID (12 bits) allows up to 4094 VLANs.

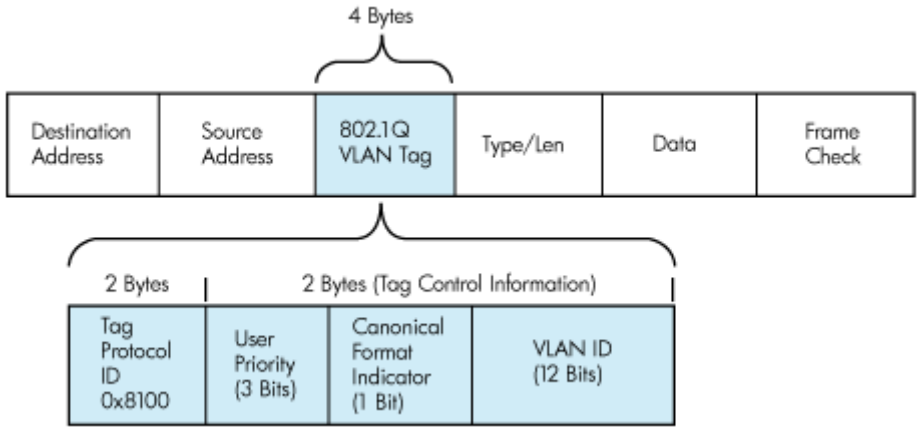


Figure 5. 802.1Q VLAN tag with 802.1p priority field

Regarding the current adoption of Ethernet technology by the automotive industry, standard Ethernet is not considered as a solution. Due to EMC issues, shielded cables have to be used, resulting both in higher cost and weight. This is the reason that the automotive industry considers the unshielded single twisted pair cable instead. OPEN Alliance is a special interest group consisting of manufacturers and OEMs. Its purpose is to promote the "wide scale adoption of Ethernet-based, single pair unshielded networks as the standard in automotive applications" [10]. BroadR-Reach is the Ethernet solution that is promoted by the OPEN Alliance and is expected to be used soon in vehicles [11]. Unfortunately this technology is proprietary. IEEE is working on a Gb/s automotive version, which will be non-

proprietary. For this reason the Reduced Twisted Pair Gigabit Ethernet study group [12] was created in order to support 1 Gb/s operation in automotive and industrial environments. This standard is expected to become available around 2018.

2.4 AUTOSAR

AUTomotive Open System Architecture is a software architecture standard developed jointly by automotive manufacturers, OEMs and tool developers. This standard was created to satisfy the need for standardization of basic software and the interfaces to applications/bus systems, since future E/E architectures will have a common basis across OEMs [13]. The intention behind this effort was to reduce system complexity and keep the development cost feasible. Some additional goals of AUTOSAR include the scalability across different vehicles and platforms, maintainability throughout the product lifecycle and the sustainable utilization of natural resources [14].

AUTOSAR follows a layered architecture, where hardware, basic software, runtime environment and application software are separated from each other [15]. The basic concepts of AUTOSAR are the following:

- Software Component (SWC) Each SWC should be assigned to one ECU and encapsulates part of the functionality of the application [16]. The component's implementation is independent of the infrastructure following the basic design concept of separation between layers.
- Runtime Environment (RTE) The RTE provides a communication abstraction to the SWCs connected to it, providing the same interface and services both for inter and intra ECU communication. Since the requirements of SWCs running on RTE may vary, different ECUs may have different RTEs.
- Basic Software (BSW) The BSW is essential to run the functional part of the software. It is the standardized software layer, that provides services to the SWCs [12]. It contains both standard and ECU specific components.

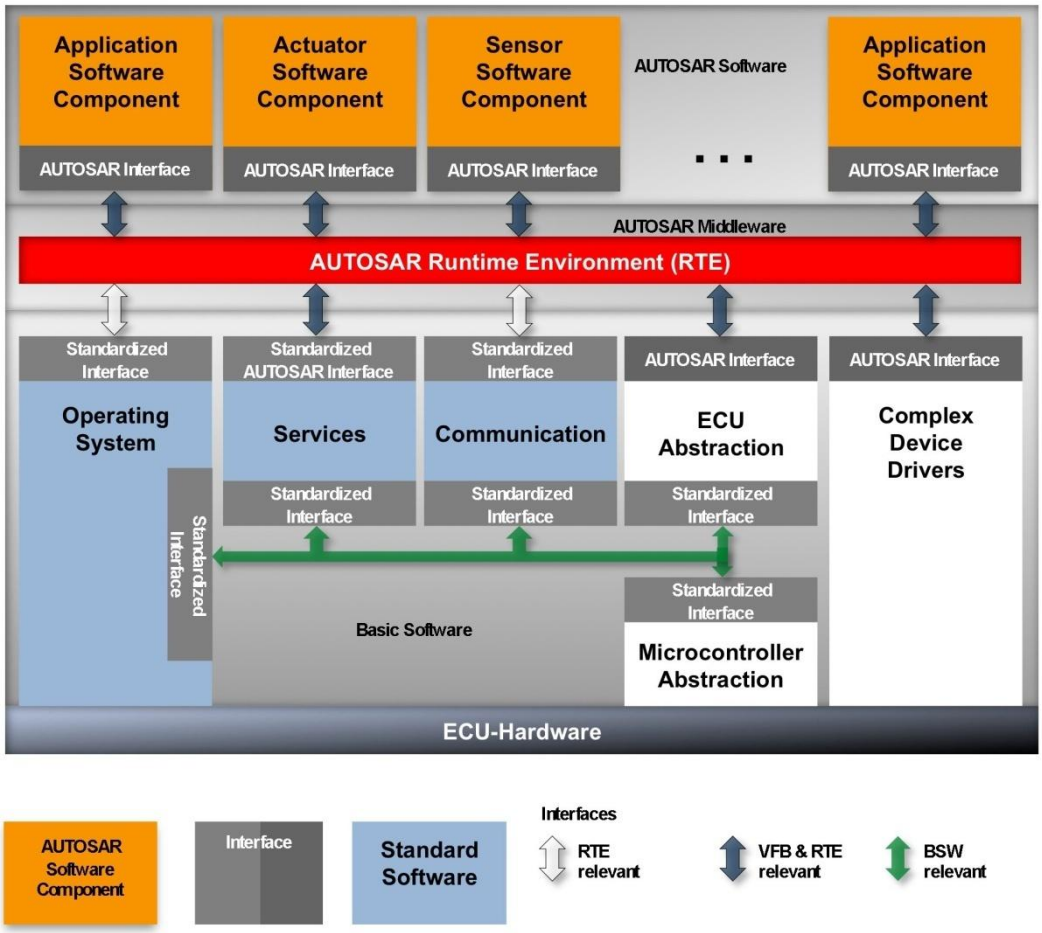


Figure 6. AUTOSAR layered architecture [10]

2.5 Network Calculus

Network Calculus is a well-established technique used for performance analysis of real-time communication networks. It was originally proposed by Cruz [17, 18] and further developed by Le Boudec and Thiran [19]. The foundation of network calculus is based on the min-plus algebra and takes into consideration networks consisting of service nodes and the packet flows between them. Network calculus is a theory for deterministic queuing systems. In contrast with traditional queuing theory, which models the system's behavior based on stochastic processes and probability functions, network calculus applies bounding constraints on packet arrival and service [20]. This enables delay and backlog bounds to be calculated. This property makes network calculus very attractive for real-time applications which require QoS guarantee and has already been used for industrial and avionics network applications.

The building blocks of network calculus are the arrival and the service curve of a node as well as two functions of the min-plus algebra, the convolution and the deconvolution represented by the operator $\oplus$ and $\ominus$ respectively.

Convolution

F denotes the set of wide sense increasing sequences or functions such that for $t < 0$, $f(t) = 0$. For two functions f and g belonging to F, the convolution is defined as:

$$(f \oplus g)(t) = \inf_{0 \le s \le t}\{f(t-s) + g(s)\}$$

(If $t < 0$, $(f \oplus g)(t) = 0$)

The infimum of a subset *S* of a partially ordered set *T* is the greatest element of *T* that is less than or equal to all elements of *S*, denoted by *inf*.

Deconvolution

The deconvolution of the two functions is:

$$(f \ominus g)(t) = \sup_{u \ge 0}\{f(t+u) - g(u)\}$$

If both $f(t)$ and $g(t)$ are infinite for some t , the above equation is not defined. In contrast with min-plus convolution $(f \ominus g)(t)$ is not necessarily zero for $t \le 0$.

The supremum of a subset *S* of a totally or partially ordered set *T* is the least element of *T* that is greater than or equal to all elements of *S*, denoted by *sup*.

Arrival Curve

If $R(t)$ is the cumulative function, expressing the amount of data arriving in a time interval $[0, t]$, then a flow entering a node has an arrival curve $a(t)$ if for all $0 \le s \le t$,

$$R(t) - R(s) \le a(t-s)$$

where $a(t)$ is a non-negative and non-decreasing function.

As shown by Le Boudec and Thiran at [15], the leaky bucket controller concept can be used as a regulation method denoting the constraints of the traffic. This controller constrains a flow by an affine curve $b(t) = \sigma + \rho t$, where σ is the maximum amount of data that can arrive in a burst and ρ an upper bound on the long term average rate of the traffic flow. This expression is similar to the concept of burstiness constraint as described by Cruz [13]. If R(t) is the instantaneous rate of data arriving on time t , then $\forall x, y; y \ge x, x \ge 0$ the burstiness constrained is:

$$R \sim b \Leftrightarrow \int_{x}^{y} R(t)dt \leq \sigma + \rho(y - x)$$

Service Curve

A service curve describes the service of a flow F inside a node. Let $R^*(t)$ be the amount of data output in a time interval $[0, t]$. Then the non-negative and non-decreasing function $\beta(t)$ is called the service curve for flow F in this node if for all $t \geq 0$,

$$R^*(t) \geq R \oplus \beta(t)$$

Output Characterization

For a flow constrained by an arrival curve $a(t)$, traversing a system that offers a service curve $\beta(t)$, the output flow is constrained by the arrival curve [21]:

$$a^*(t) = a(t) \ominus \beta(t)$$

Pay burst only once

As shown in Figure 7, a flow with an arrival curve $a(t)$ enters the first node which offers a service curve $\beta_1(t)$. The output flow will be constrained by the arrival curve $a^*(t) = a(t) \ominus \beta_1(t)$. This output flow will be the input flow of the second node. Likewise the output flow of the second node will be constrained by the arrival curve $a^{**}(t) = a^*(t) \ominus \beta_2(t)$. The total delay of the system is the sum of the delays of each node.

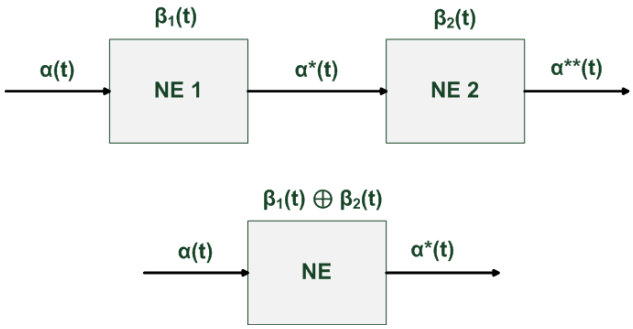


Figure 7. Concatenation of nodes

The service curve offered to a flow that traverses two nodes in sequence is $\beta(t) = \beta_1(t) \oplus \beta_2(t)$ (Concatenation of Nodes) [15]. By applying the network service curve in order to calculate the system's delay, the results obtained are tighter. This happens because in the first case the delay due to the burstiness of the flow was calculated twice. This is referred to as "pay burst only once" phenomenon.

Applied network calculus

As mentioned earlier network calculus has been successfully applied in the field of industrial automation. The most significant research in this field has been performed by Georges et al. [22-25]. In [18], inspired by the work of Cruz, the authors proposed a model of a switch based on a sequence of elementary components. Their goal was to obtain a formula for the calculation of end-to-end delay in switched Ethernet networks. In [19], the authors make a comparison of different theoretical models of Ethernet switches. A network simulation tool was used in order to identify the one that represents more accurately the functionality of a switch. In a later work [20], a formal method to guarantee a deterministic behavior of switched Ethernet was introduced. The previously described models were extended taking into consideration the QoS mechanism which enables prioritization according to IEEE 802.1p. Finally in [21], using the new model of the switch, comparison between strict priority and weighted fair queuing scheduling policies was performed. Also, new formulas that calculate the upper bound of the maximum end-to-end delay were presented.

Network calculus has also been applied in the automotive domain. To be more specific in [5] Rahmani et al. compared different topologies of in-vehicle networks regarding QoS performance. In [26], the authors presented an analytical model for resource planning and performance evaluation of the in-vehicle network, using network calculus theory.

2.6 Implementation Choices

After completing the literature review and having acquired a background in the research problem under investigation, we made certain choices regarding the design of the thesis project. The most suitable topology for an in-vehicle communication network was considered to be the double star topology. The reason behind this choice is that this topology is the most viable to be implemented by an automotive manufacturer. The two switches can be placed in the front and the back of the cabin of the car, allowing a more flexible design. In this way, domain masters that exchange messages more frequently can be connected to the same switch, achieving better network performance. Moreover, other advantages of this topology are the network resilience in case a node fails and the flexibility in choosing link capacities for separate parts of the network [5].

The strict priority policy was selected as the queuing policy. Since the traffic of an in-vehicle network varies both in terms of requirements but also in criticality level, specific QoS guarantees have to be provided for each type. The strict priority policy reflects more

accurately this requirement and is the most straight forward solution. The traffic of the network was grouped into four types and each one was assigned to its respective priority level. Another factor contributing to this decision is that in practice switches only support up to four priority levels.

Since the double star topology adds the extra cost of the two switches, it is very important for automotive manufacturers to be able to evaluate the potential of commercial of-the-shelf solutions. This is why such switches were used and not a high-end solution. Another reason behind that choice was to demonstrate that the proposed solution is feasible with existing technology.

Another important design choice was the use of AUTOSAR as an operating system for the development boards of our experiments. In this way, our system becomes more realistic and complies with the automotive standards. In addition to that the development boards used were equipped with a Freescale MPC5567 processor which is typical for automotive applications.

3. Methodology

In order to evaluate the end-to-end delay, both theoretical and experimental methods were deployed. End-to-end delay is defined as the time it takes for a frame to traverse the backbone once it is received from an input port of the first switch until it is fully transmitted by an output port of the second switch. As a first step, a theoretical model that accurately reflects the traffic load and patterns, as well as timing constraints (specified by Volvo Car Corporation) was created. Then, an experimental setup which was based on the theoretical model was produced. In this way the results obtained from the theoretical analysis can be compared with those measured from the experiments.

3.1 Use Cases

The following use cases were investigated:

- **Use case 1.** The traffic load for the first two types of traffic complies with the existing load of an in-vehicle network of a Volvo car. The third priority traffic (video traffic) has a rate of 32Mbps and the best effort traffic has a rate of 8Mbps.
- **Use case 2.** To investigate the effect of increased video traffic, the rate of third priority traffic was doubled reaching 64Mbps. All other traffic remains the same as in use case 1.
- **Use case 3.** In the future it is expected that the amount of control data exchanged within the network will be significantly increased. Therefore, in this use case the payload of the first priority traffic is doubled. All other traffic remains the same as in use case1.

The three use cases were examined both for a standard MTU of 1500 bytes, as well as for an MTU of 800 bytes.

3.2 Traffic Model

For the model of our system the following four priority classes will be employed:

- **Hard real time messages:** This class has the highest priority and consists of control messages regarding the powertrain, chassis and body electronic systems. These are small sized periodic messages. Usually they have more than one destination (multicast). No packet loss is tolerated in this priority class.

- **Soft real time messages:** This class has second priority and consists of diagnostic messages. These are also small sized messages that exhibit periodic behavior. However they are not as highly critical as the previous class and an amount of missed deadlines can be tolerated.
- **Video data:** This class has the third priority and consists of video streams. One or more cameras will stream video of MJPEG format. This class generates a lot of traffic since it has high data rate. Certain losses ≤0.1% can be tolerated [5].
- **Best effort data:** Finally the last class consists of best effort messages. Typical applications can be web browsing or file transferring. These applications use TCP which facilitates mechanisms that ensure reliable transfer of data. This class consists of non-critical traffic that has no real time constraints. Since TCP is used, any packets lost will be retransmitted.

In our model strict priority policy will be deployed. This means that any frame of a given priority will be served only if there are no other frames of higher priority present in the queues. Therefore the service offered by the switch is not the same for all priority classes. The traffic generated by each class will be referred to as flow *i*, where *i* is the respective priority. The arrival curves of the flows will be expressed as $a_i(t) = b_i + r_i t$ (where b is the burst size and r is the average rate of the flow), following the leaky bucket controller concept. As shown by Georges et al [25], the service curve offered to each flow can be expressed as $\beta(t) = R * (t - T)^+$, where R is the rate at which data will be served expressed in bits/sec and T is the total delay suffered by a flow before being actually served. T may consist of two factors depending on the priority level of the flow. The first one is the interference from higher priority flows being served at that point and the second one is the blocking point due to a non preemptive transmission of lower priority flows. In detail the service curve for each priority can be expressed as:

- **1st Priority:** The packets of this priority will be selected first. However since forwarding of a packet is non-preemptive, the packets of this flow might be blocked by a lower priority packet that is being transmitted. The service curve of this flow is:

$$\beta_1(t) = R * (t - T)^+$$

$$T = \max\{L_{2,max}, L_{3,max}, L_{4,max}\} / C \ , R = C$$

- **2nd Priority:** The packets of this priority have to wait in case any packet of flow 1 has to be served (interference). Furthermore as mentioned earlier due to the non-preemptive transmission of packets, the packets of this flow can again be blocked by a lower priority packet from flow 3 or 4. Since flow 1 will be always served first the

forwarding rate left for flow 2 will be limited to $C - r_1$. The service curve of this flow is:

$$\beta_2(t) = R * (t - T)^+$$

$$T = \frac{b_1}{C - r_1} + \max\{ L_{3,max} , L_{4,max} \} / C \ , R = C - r_1$$

- **3rd Priority:** The packet of this priority can only be served if all higher priority packets have already been served and no packet of flow 4 is being transmitted at that point of time. The service rate is limited to $C - r_1 - r_2$. The service curve of this flow is:

$$\beta_3(t) = R * (t - T)^+$$

$$T = \frac{b_1 + b_2}{C - r_1 - r_2} + \frac{L_{4,max}}{C} \ , R = C - r_1 - r_2$$

- **4th Priority:** The packet of this priority can only be served if all higher priority packets have already been served. The service rate is limited to $C - r_1 - r_2 - r_3$. The service curve of this flow is:

$$\beta_4(t) = R * (t - T)^+$$

$$T = \frac{b_1 + b_2 + b_3}{C - r_1 - r_2 - r_3} , R = C - r_1 - r_2 - r_3$$

3.3 Switch Model

In traditional CSMA/CD Ethernet, a collision can cause a random back-off of the node before retransmitting. Therefore the calculation of an upper bound for the delay is not possible. In full duplex switched Ethernet collisions are completely avoided. However this new element (the switch) introduces delays in the network traffic, whose upper bounds can be calculated in a deterministic way.

The total delay of a frame passing through a switch can be expressed as [27]:

$$D_{switch} = D_{proc} + D_{queue} + D_{trans} ,$$

where D_{proc} is the processing delay of a frame by a switch. It depends on the hardware capabilities of the switch, often referred to as switch fabric. The value of this delay lies in the area of a few microseconds and is often provided by the manufacturer or can be found

experimentally. For the needs of the thesis the switch fabric was measured experimentally[1] and was found to be 10μs (in the worst case). Measurements were performed both for a maximum sized Ethernet frame as well as for frame size equal to those of the control data. The worst case was observed for the small frame size as it can be seen in Figure 8.

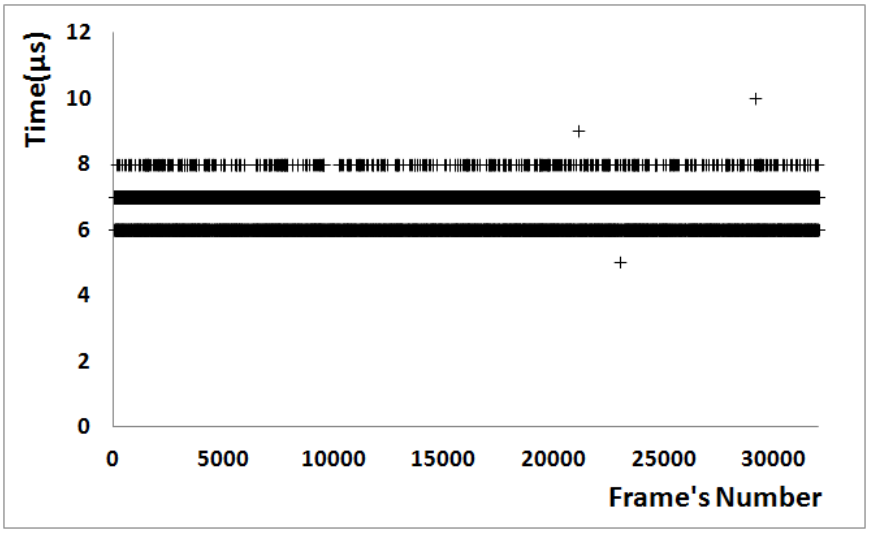


Figure 8. Switch Fabric measured values

D_{queue} represents the delay caused by the queuing of the frame before being transmitted by the output port. This type of delay has the greatest impact on the total delay imposed by the switch. Finally, D_{trans} is the time needed for the frame to be transmitted. This can be easily calculated by the expression

$$D_{trans} = \frac{L}{C} ,$$

where L is the frame length and C is the maximum link capacity.

Figure 9 shows a conceptual model of the switch. A frame is received by Rx port and is then processed by the switch fabric and placed in the appropriate priority queue. The queues although being represented as discrete logical elements they are located in the same physical shared memory. The queue scheduler decides which frame will be transmitted based on the queuing policy. The frame is then transmitted through the Tx port.

In case high bit rate traffic arriving from multiple ports is destined to one output port, the rate at which traffic will be forwarded by the output port is limited by the link capacity. As a consequence frames have to be stored in buffers located in the shared memory of the switch. If the shared memory becomes full then any received frame, regardless of which port it was received or to which priority it belongs to, will be dropped. Packet drops are not

[1] The details of the experiment are presented in the Appendix A.

acceptable for real time communications. Therefore the calculation of buffer sizes as well as the utilization of shared memory should also be examined. As mentioned earlier by applying network calculus, theoretical upper bounds can be derived both for the queuing delays and the buffer sizes.

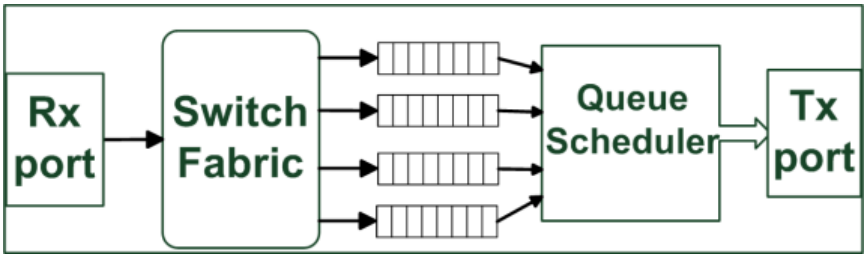


Figure 9. Switch Model (Notice the four queues for the traffics having different priorities)

3.4 System Adaptation

The system traffic, as shown in Figure 10, represents the worst case scenario and is modeled into four flows. As mentioned earlier our system has a double star topology. In such topology the frames that will experience the highest delay are those traversing through both switches. Therefore the traffic generated on one side of the network destined for the other side of the network will be examined (from left to right). In order to have realistic measurements the system traffic was fully loaded on both directions. The scenario under examination represents the worst case because high volume traffic of all priorities is generated on the left side of the network and traverses switch A through the same output port, which is the bottleneck of the system. The highest amount of data is generated from the Video Source and the FTP server.

Flow $a_1(t) = a_1^{PCD}(t) + a_1^{BED}(t) + a_1^{SD}(t)$ is the sum of all flows of 1st priority messages generated from the powertrain and chassis domain, body electronics domain and safety domain. Flow $a_2(t) = a_2^{PCD}(t) + a_2^{BED}(t) + a_2^{SD}(t)$ is the sum of all flows of 2nd priority messages generated from the powertrain and chassis domain, body electronics domain and safety domain. Flow $a_3(t)$ is the flow of 3rd priority video data destined for the video sink. Finally, flow $a_4(t)$ is the flow of 4th priority best effort data sent from an FTP server to a client.

Each of the four types of messages is placed in its respective queue by the switch fabric. The queue scheduler of the transmitting port will forward the frames according to the scheduling policy. The output traffic of each flow has an arrival curve of $a_i^*(t) = a_i\ (t) \ominus \beta_i(t)$. These curves will be the input for switch B. However, in switch B, traffic exchanged

between the communication domain and the infotainment domain must also be taken into consideration. Furthermore the incoming traffic to switch B has multiple output destinations. The video traffic is directed to the port connected to the video sink, no other types of traffic arrive at this port so it only has one queue. The same applies to the best effort data flow. For the two domain masters residing on this side of the network there are two queues at each output port and the arrival curves for each flow are expressed as:

$a_1'(t) = a_1^*(t) + a_1^{ID}(t)$ and $a_2'(t) = a_2^*(t)$ for the communication domain

$a_1''(t) = a_1^*(t) + a_1^{CD}(t)$ and $a_2'(t) = a_2^{CD}(t)$ for the infotainment domain

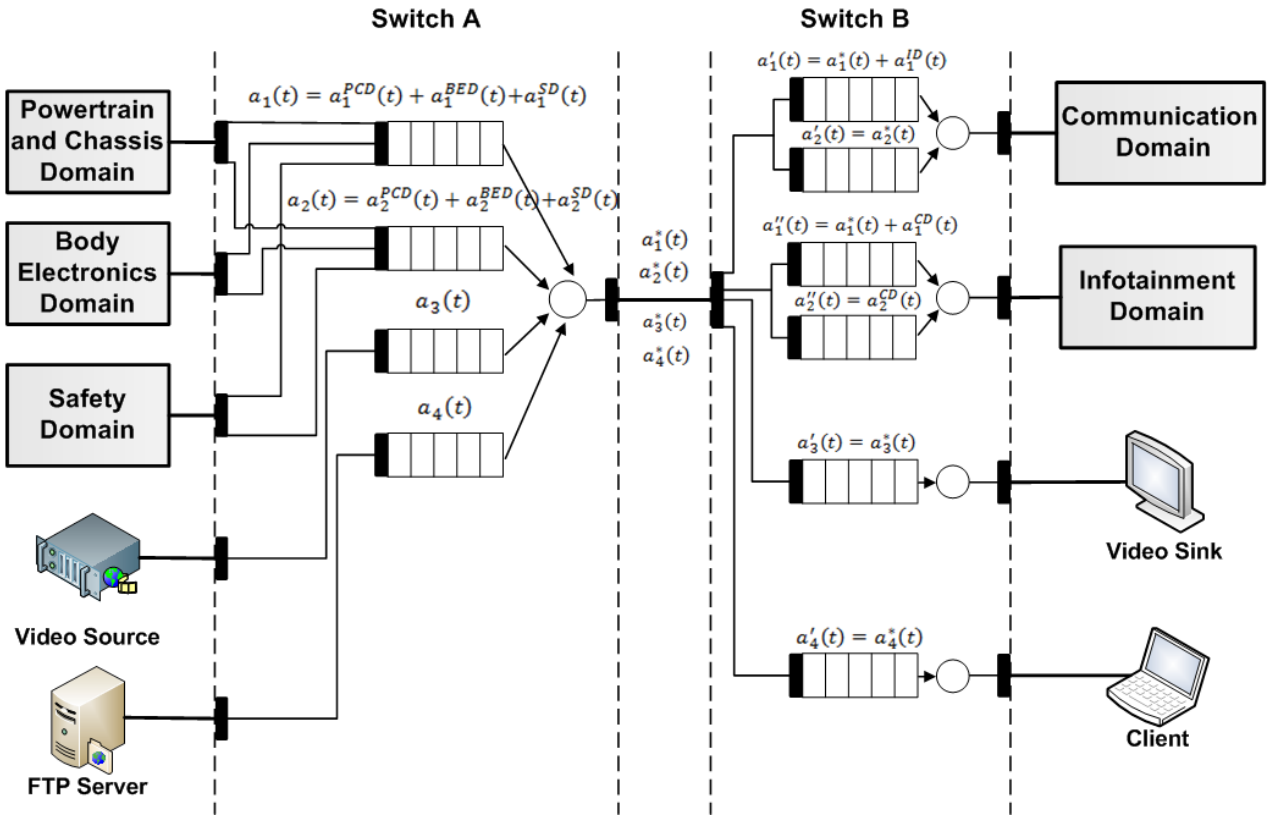


Figure 10. System traffic model

FlexRay is used nowadays as the backbone for in-vehicle communication by some automotive manufacturers. In order to evaluate Ethernet as an alternative solution, the current traffic load of the FlexRay implementation of Volvo Car Corporation was adapted to a switched Ethernet implementation.

Each node sends a given number of frames to one or more other nodes. These frames are transmitted at their predefined time slot in the FlexRay cycle. The length of the cycle is 5ms and a frame may have a period of n*5ms, where n=0,1,2,… . It was observed that a node

does not transmit all of the frames at once (sequential time slots), but there is a time gap between each transmission. This behavior is represented in our model. The frames of each node will be transmitted with a given offset between them (200μs) while the periods remain the same.

For the first two priority classes of the system under examination, the exchanged messages are of periodical nature. The traffic generated on the vehicle's backbone network is known beforehand both in terms of size and period. Therefore the arrival curve can be expressed as $a(t) = b + rt$, where b is the maximum frame size of the flow[2] and r is the average data rate. This has the following advantages. First of all a(t) is a deterministic upper bound of the specific traffic flow, as shown in Figure 11. However, the data arrival on a physical network is limited by its link capacity C. As a consequence the arrival curve will be expressed as :

$$a(t) = \min\{Ct, b + rt\}$$

Each of the frames sent periodically by a node, may represent a flow with an arrival curve $a(t)$. The arrival curve of the total traffic of a node can be found by simply adding the arrival curves for each frame flow. This however leads to a large burst value. By introducing this "mimicking FlexRay" behavior, only one frame will be transmitted at a time by a node, thus keeping the burst size to a minimum of one frame size. This approach results in a tighter estimation of end-to-end delays and backlogs.

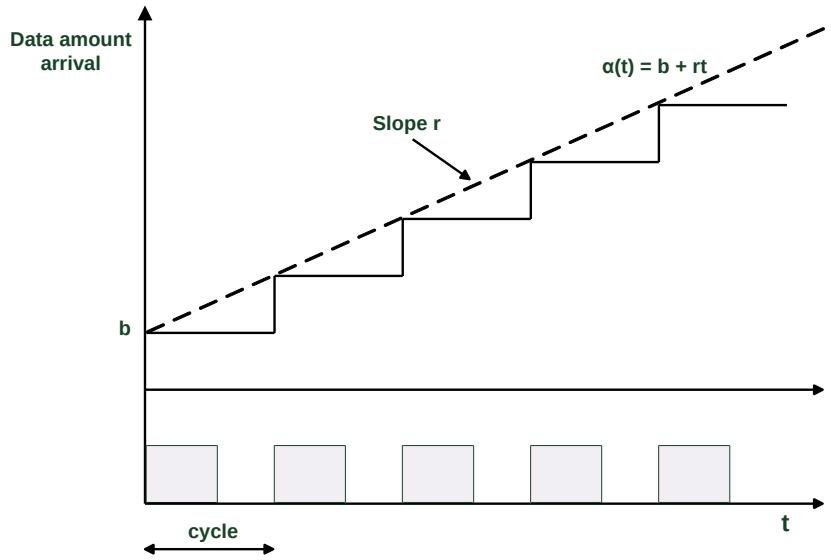


Figure 11. Arrival Curve of periodic traffic

[2] The frame size for the first and second priority traffic is 90 bytes (32 bytes of payload plus overhead of additional layers UDP,IP and Ethernet).

In order to analyze the entire system in a deterministic way the last two priority classes must also have arrival curves constrained by a leaky bucket. Indeed the third priority class which consists of MJPEG video data can be modeled in the same way. MJPEG format is preferred by the automotive manufacturers due to its low complexity and low hardware requirements. Each frame is a JPEG picture which is compressed individually and then sent to the destination. A typical configuration consists of a resolution of 640X480 (standard VGA) and a rate of 30 fps. The arrival curve of this flow can be expressed as $a(t) = b + rt$ where b is the frame size of a JPEG picture and r the average data rate.

Finally the traffic of the fourth priority class is random by nature. Therefore traffic shaping has to be applied at the source so that the arrival curve of this flow will conform to a leaky bucket curve expression. In this way determinism can also be achieved for this class. A simple leaky/token bucket traffic shaper can be applied. The numerical expressions for the arrival curves of the system traffic are:

$a_1\ (t) = 270 + 297000t$

$a_2\ (t) = 270 + 216000t$

$a_3\ (t) = 138173 + 44281981t$

$a_4\ (t) = 8192 + 1000000t$

3.5 Delay and backlog calculation

For a flow with an arrival curve $a(t)$ traversing through a system with service curve $\beta(t)$, Le Boudec and Thiran have shown that the maximum delay D is the maximum horizontal distance between the two curves and the maximum backlog B is the maximum vertical distance between the two curves (Figure 12). The inflection point g is defined as the point where the arrival curve changes slope [28]. That is the point where $= b + rt$. Therefore, $g = \frac{b}{C-r}$

For the calculation of the backlog two cases are distinguished. For $T \leq g$ the maximum backlog is found at point *g*, while for $T > g$ the maximum backlog is found at point *T*. The derived expressions for the calculation of the backlog and delay of a given flow are the following:

$$B = b + rg - R * (g - T) \qquad ,for\ T \leq g$$

$$B = b + rT \qquad ,for\ T > g$$

$$D = \frac{b + rg}{R} + (T - g)$$

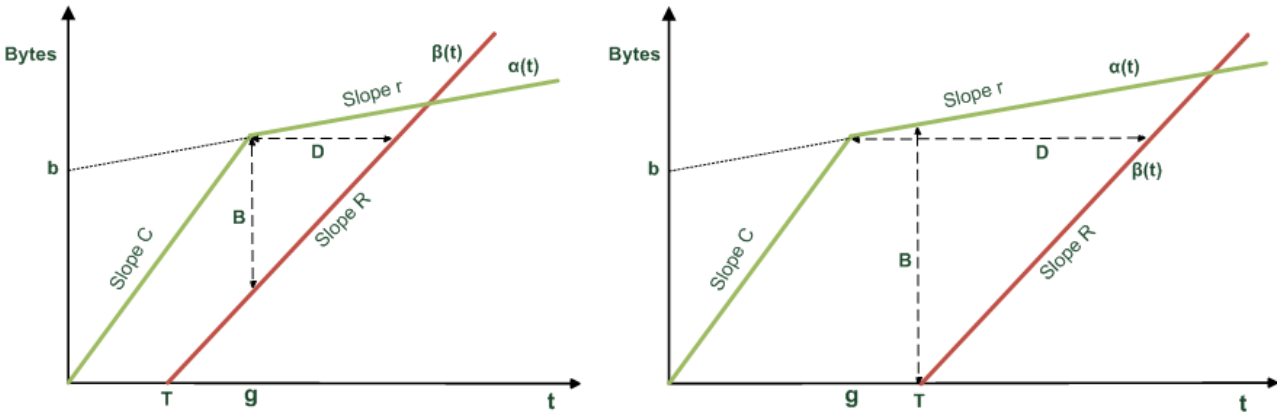


Figure 12. Backlog and Delay illustration for T≤g and for T>g

For the calculations of the delay and backlog values, a program was developed in Java. The source code is presented in Appendix B.

3.6 Experimental Setup

The following equipment was used in order to perform the experiments:

1. Netgear GS108Tv2 smart switch (two items).
2. QRTECH ODEEP development board (five items).
3. iSYSTEM winIDEA iC3000 debugger.
4. Fluke networks inline TAP-100 (two items).
5. Vector VN5610 network interface.
6. Laptop computers with Linux operating system (four items).
7. Desktop computer with CANoe by Vector.

In order to measure the end-to-end delay of a frame, the frame has to be time stamped while entering and exiting the network. Then by subtracting the two time values the end-to-end delay can be calculated. For the measurement of the first priority message delay, one network tap was connected on the link between the powertrain and chassis domain and switch A and the other tap was connected to the link between switch B and the infotainment domain. This was done since the rate of data traversing that path is the highest for the system. Due to the fact that all diagnostic messages are destined to the communication domain master, the second priority message delays were measured between the powertrain and chassis domain and the communication domain. The two taps were then connected to an experimental computer through the VN5610 network interface, which captured and time-stamped the frames. The trace of the captured traffic was analyzed using CANoe. The network is full duplex and special caution has to be exercised in capturing the desired direction of traffic.

The same approach was applied to the third priority flow between the video source and sink. The fourth priority flow is not considered since it is best effort traffic.

The development boards were running ArcCore's Autosar Implementation [29] with an IP Stack. Support for 802.1p and 802.1Q in Autosar is specified for version 4.1, which is not released by any developer during this thesis. Therefore, the IP Stack was extended to support these protocols. Figure 13 illustrates the experimental setup.

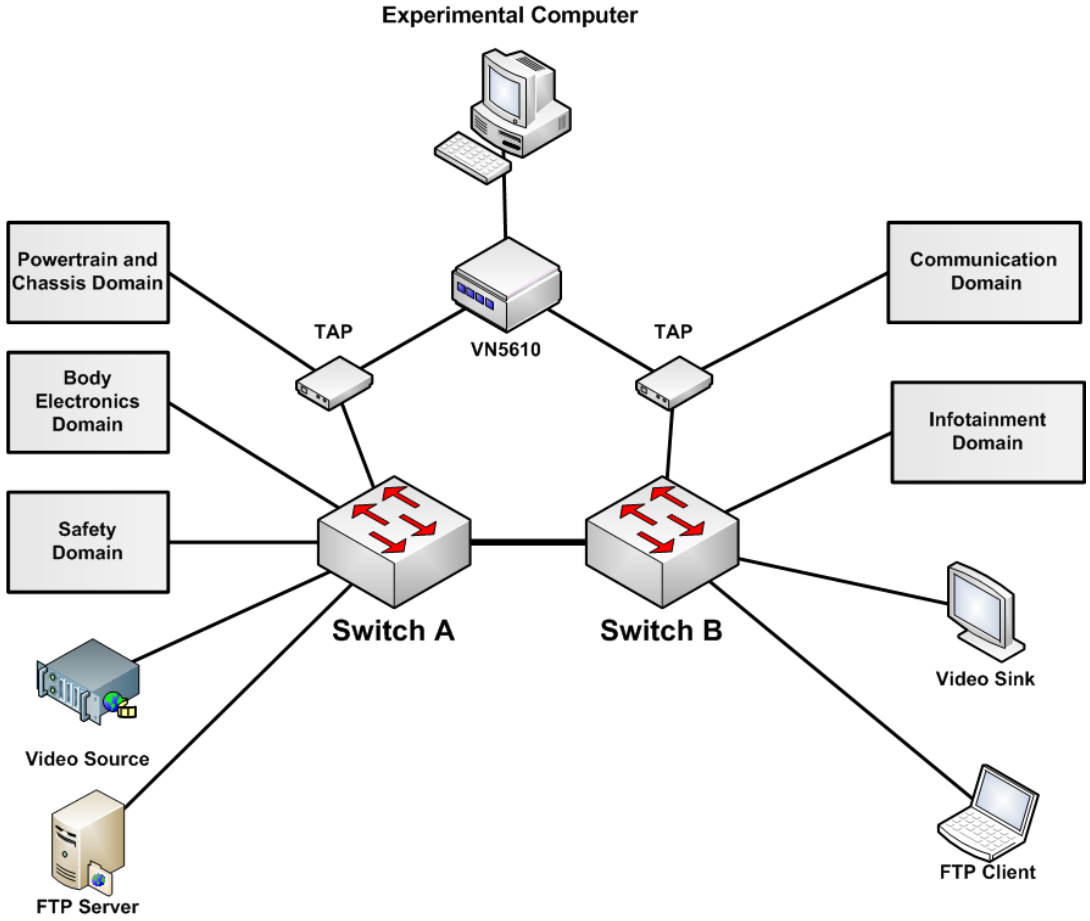


Figure 13. Experimental Setup

3.7 Implementation Details

The specifications defined that frames transmitted by a domain had multiple destinations (multicast). In order to implement this, the network was divided into VLANs. Each domain belonged to a specific number of VLANs. Two separate VLANs were also created, one for the video traffic and one for the FTP traffic. The use of VLANs allows the logical isolation of traffic within the same physical network. This feature is beneficial in terms of resilience, isolating a "babbling idiot" into its respective VLAN. Also this is beneficial in a security perspective, in case an attacker gains access i.e. to the FTP VLAN; he

will not have access to the rest of the network. An alternative for implementing multicast in such a network is the IGMP protocol. However the switches must support layer 3 functionality.

The following actions were taken to ensure behavior according to the IEEE 802.1Q and 802.1p. The development boards, representing the domain masters, were configured to tag the transmitted frames with the appropriate VLAN-id and priority. The development boards were programmed to discard frames upon reception since no useful data were transmitted. The appropriate VLAN configuration was applied to the switches as well as the mapping of the PCP value with the priority queues of the switch. The link between the switches was the trunk link, on which traffic from all VLANs was allowed.

The laptop used as video source streamed an MJPEG video file over UDP using the VLC media player. The video sink was playing the stream using the same software. The FTP client was retrieving a large file form the FTP server, on which traffic shaping had been implemented. As mentioned earlier, all four laptops were running Linux. A typical example of the commands used to configure them is presented below.

```
#for VLAN TAG
vconfig add eth0 17
ip addr add 192.168.0.30/24 dev eth0.17
ifconfig eth0.17 up

#for priority TAG
iptables -t mangle -A POSTROUTING -d 192.168.0.31 -j CLASSIFY
--set-class 0:4
vconfig set_egress_map eth0.17 4 4

#rate control
tc qdisc add dev eth0 root tbf rate 8mbit burst 8kb limit 32kb
```

The switches used were gigabit switches. Our network operated with a rate of 100Mbps. For some of the switch's ports the auto-negotiation feature had to be disabled and the data rate had to be configured manually. If one device on the link is configured to auto-negotiate and the other is not, an auto-negotiation failure occurs, leading to duplex mismatch. As a result the auto-negotiating device will operate in half duplex by default. This mode of operation leads to degraded performance and possible collisions.

3.8 Data processing

The captured traffic was stored in a binary log file in CANoe. CANoe however, did not provide all the features for statistical analysis that were required for this thesis. Using a

CAPL script the valuable information was extracted from the binary log files and was converted into ASCII text files. These files were imported to Microsoft EXCEL for further processing.

By applying the appropriate filters, the matching time values[3] of the same frame were sorted into two columns. The two time values were subtracted and the delay of each frame was obtained. The precision of the measurement is 1μs. The same procedure was repeated for all priorities in all use cases. The values were also used to produce charts and diagrams which are presented in section four. Tables with minimum, maximum and average delays per priority were also created.

[3] The time-stamp value of a frame entering the network and the time-stamp value of the same frame leaving the network.

4. Results and Discussion

In this chapter both theoretical and experimental results will be presented. Initially, the calculated theoretical values regarding the end-to-end delay and backlog are shown in Tables 1 and 2. The backlog value displayed is the total backlog of the output port of switch A. This is the bottleneck of the system, which has the highest requirements for buffer. As it can be seen from the tables below the highest amount of memory needed is in case 2 for an MTU of 1500. The backlog value in that case is 125KB, while the switch used in the experiments has a total amount of 128KB of shared memory. Therefore, no packet drops should occur for any of the use cases. The calculated values are pessimistic and the actual requirement for memory is significantly lower.

Table 1. Theoretical Delay and Backlog values (MTU 1500)

MTU 1500			
Flow	**Case 1**	**Case 2**	**Case 3**
1st	329 μs	329 μs	351 μs
2nd	377 μs	377 μs	408 μs
3rd	1287 μs	3476 μs	1454 μs
4th	18904 μs	80937 μs	19195 μs
Backlog	38KB	125KB	41KB

Table 2. Theoretical Delay and Backlog values (MTU 800)

MTU 800			
Flow	**Case 1**	**Case 2**	**Case 3**
1st	216 μs	216 μs	237 μs
2nd	259 μs	259 μs	287 μs
3rd	1058 μs	3247 μs	1224 μs
4th	18719 μs	80527 μs	19008 μs
Backlog	37KB	124KB	39KB

To continue with, the experimental results for the first use case are presented below. In Figure 14 it can be observed that all measured values are below the theoretical bound. Furthermore according to Figure 15, more than 50% of all measured delays experience the minimum delay for this type of traffic. The minimum delay value of 28μs does not contain any delay due to queuing. The area of delay values between 29 and 90μs describes the case where a frame is blocked by an ongoing non-preemptive transmission of a frame. The frames in the area of delay values between 90 and 210μs, suffer high queuing delays due to blocking from lower priority traffic and represent 41% of the total measured values. Finally only 0, 32% of the frames had a delay value higher than 210μs.

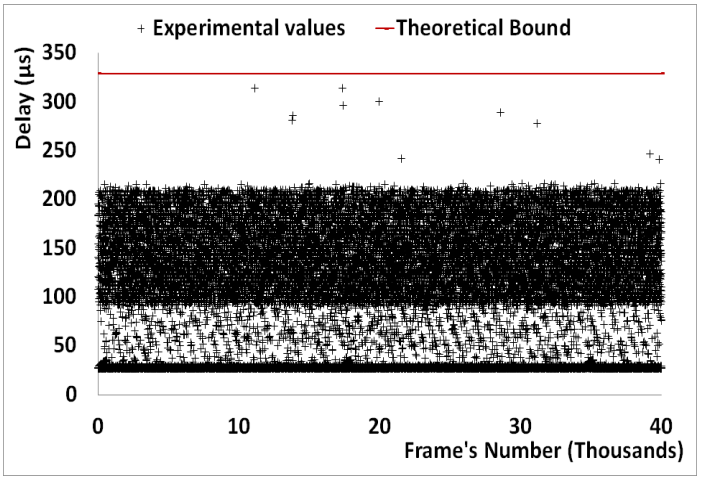


Figure 14. Case 1 first priority delay values

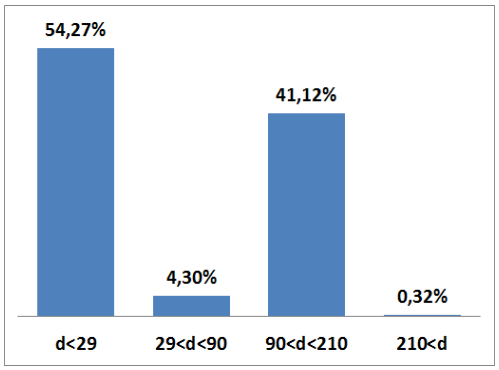


Figure 15. Case 1 first priority delay spread

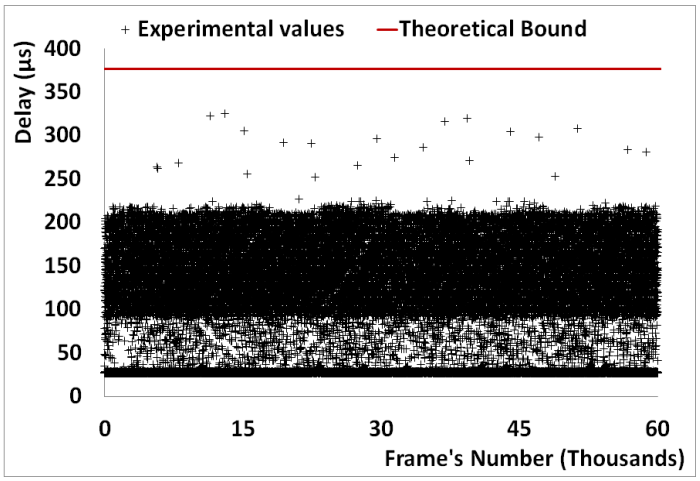


Figure 16. Case 1 second priority delay values

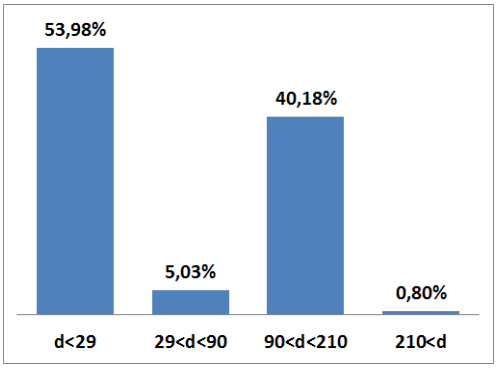


Figure 17. Case 1 second priority delay spread

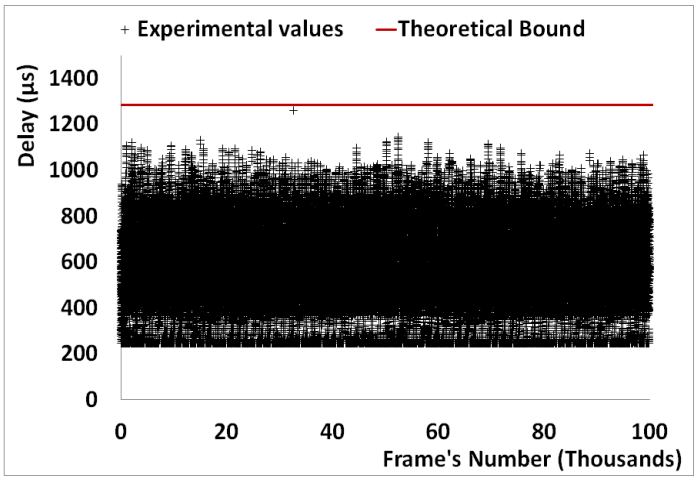


Figure 18. Case 1 third priority delay values

The experimental values of the second priority traffic are presented in Figure 16. Again, the measured values are below the theoretical bound and the spread of the values follow a similar pattern as in the first priority traffic. The measured delays for the third priority traffic are presented in Figure 18. There are no specific patterns and the delay values

are scattered from a minimum of 246μs to a maximum of 1263μs, with only one value being relatively close to the theoretical bound of 1287μs.

After the presentation of the first use case, only the experimental cases for the first priority traffic will be illustrated. The remaining diagrams are available in Appendix C. Only the first priority traffic will be discussed, since it is the most critical.

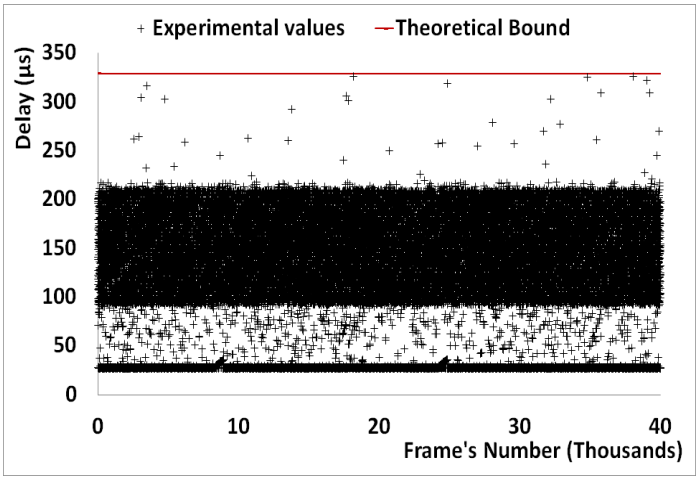


Figure 19. Case 2 first priority delay values

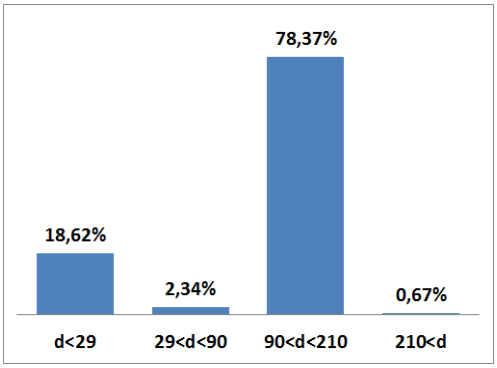


Figure 20. Case 2 first priority delay spread

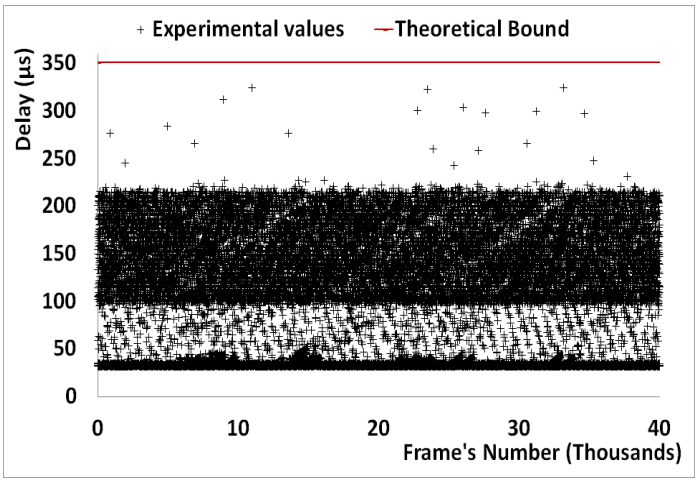


Figure 21. Case 3 first priority delay values

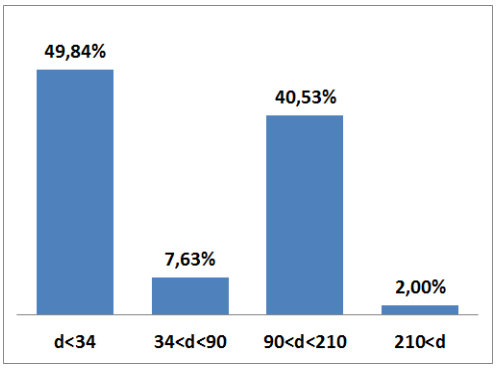


Figure 22. Case 3 first priority delay spread

The experimental values for the second use case with increased video traffic are presented in Figure 19. The theoretical bound is safe, but there are more values that come close to it, compared to the first use case. It can also be observed that 78% of the delay values are in the area of 90 up to 210μs. This was expected since the rate of the video traffic was doubled, increasing the possible number of frames that suffer from blocking factor.

The results for the third use case seen in Figures 21 and 22 are similar with those of the first use case. The only difference is the minimum delay which has now risen from 28 to 33μs, due to the fact that the frame size has increased.

The same three use cases were repeated for an MTU of 800 bytes, in order to reduce the blocking delay caused by the transmission of a maximum-sized lower priority frame. Indeed as it can be seen in all three cases both the theoretical as well as the experimental values were significantly reduced. However it was observed for the first and second case that a few experimental values slightly exceed the theoretical bound. More specific for the first use case one value has exceeded the bound of 216μs by only 3μs. For the second use case, two values exceed the bound of 216μs, the first by 10μs and the second by 1μs. For the third case no measured values were higher than the bound.

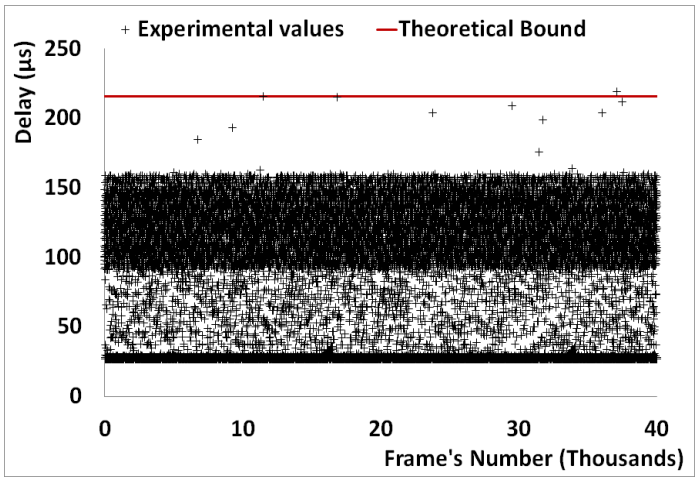


Figure 23. Case 1 first priority delay values (MTU 800)

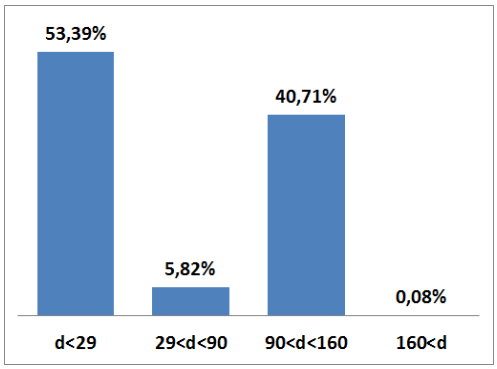


Figure 24. Case 1 first priority delay spread

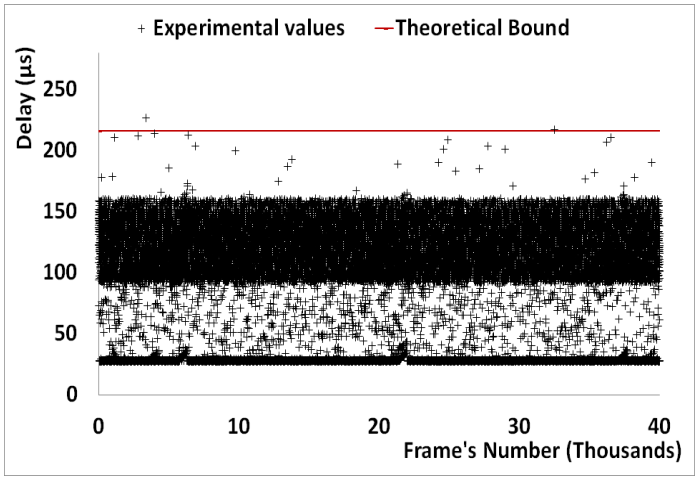


Figure 25. Case 2 first priority delay values (MTU 800)

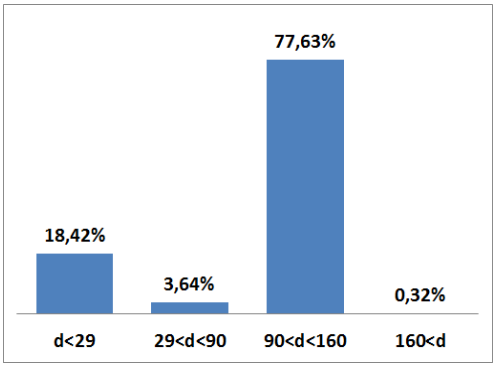


Figure 26. Case 2 first priority delay spread

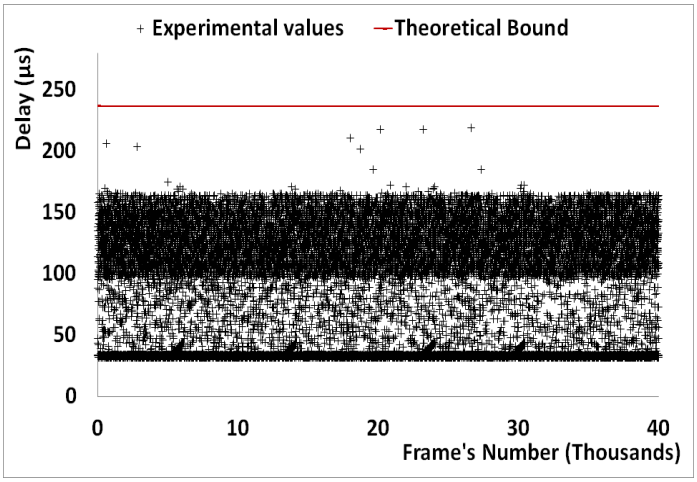


Figure 27. Case 3 first priority delay values (MTU 800)

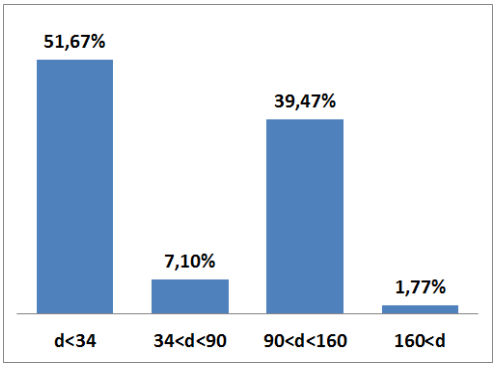


Figure 28. Case 3 first priority delay spread

In general, it can be easily observed from the diagrams that the vast majority of the experimental results are inferior to the calculus bound. Furthermore, no packet drops were observed during the experiments. Even in cases of deviation from the theoretical bound, the differences were very small. It has to be mentioned that the modeling of the system was based on a certain switch operation model. Details of the specific implementation of the switch operation were not available, as they were considered confidential by the vendor. Therefore it would be of great interest to repeat the experiment using switches of different vendors.

Although in theory a high priority frame can be blocked by one lower priority frame, during our experiments it was observed that the blocking point due to a non-preemptive transmission consists of two frames. This was used as feedback for our theoretical model resulting in higher values than originally anticipated. However, both measured and calculated delay values were considerably lower than the given 5ms deadline of the first and second priority messages.

Another point that has to be mentioned was that in case 2 for an MTU of 800 bytes, the quality of experience for the video was lower with lags observed during playback. The reason behind this, was that the video frames had to be segmented into a greater number of Ethernet frames since the MTU was 800 bytes. This increased the total delay of a video frame resulting in degraded performance. However, under situations that lower delays are required for highly critical traffic, degraded performance could be tolerated for the third and fourth priorities. For example, this can be seen as another mode of operation of the system increasing the performance for highly critical messages.

5. Conclusions and future work

The purpose of this thesis was to evaluate the use of switched Ethernet as the backbone network of an automotive system architecture. A specific system architecture for in-vehicle communication was proposed, taking into consideration the requirements provided by Volvo Car Corporation. The proposed system uses a double-star network topology. In terms of performance a single-star topology would be better; however such a topology would lead to increased cabling. On the other hand, a double star topology is easier to integrate within the vehicle.

The main objective of the thesis was to evaluate the end-to-end delay of the network and provide deterministic upper bounds for it. A theoretical model of the system under examination was produced followed by an experimental setup, which accurately reflected the system architecture. The theoretically calculated end-to-end delay values were compared with the values obtained experimentally.

The theoretical analysis was based on network calculus. Through our work, we showed that this method can be applied in the automotive domain, since the traffic generated on the vehicle's backbone network is known beforehand both in terms of size and period. This fact enables the accurate modeling of the system using network calculus. In addition to that, the "mimicking FlexRay" concept, described in section 3.4 allowed the calculation of tighter upper bounds, which otherwise would be very pessimistic.

The results of this thesis indicate that, even though Ethernet was not originally designed for real-time communication, its worst case behavior can be deterministically bounded under certain conditions. First of all, only switched Ethernet with full-duplex has to be used for such applications, since it completely avoids collisions. In this way no packet loss will occur due to collisions. However, packet loss may occur due to insufficient memory of the switch. Also, prioritization as well as logical separation of traffic traversing the system has to be performed. The standards supporting these functionalities are IEEE 802.1p and 802.1Q respectively. These standards have to be supported both by the switches as well as by the software running on the ECUs. Indeed AUTOSAR 4.1 specifies support of these standards; however, it is not yet released by any developer. The experimental results showed that Ethernet has higher end-to-end delay and jitter compared to FlexRay. On the other hand, Ethernet is able to integrate different types of traffic and is non-proprietary (open standard) which could result in more cost-efficient solutions.

An important assumption, based on which the theoretical model was produced, is the operation of the switch. As shown in 3.3, the shared memory switch architecture with output queuing was assumed, since it is the most common one. However the details of the implementation of the switch operation used in the experiments were not available. The only way to know whether the switch operated in a different way than assumed was by observation. A typical example of the previously mentioned point is the blocking point. Although in theory a high priority frame can be blocked by one lower priority frame, during our experiments it was observed that the blocking point due to a non-preemptive transmission consists of *two* frames due to the way the switch hardware dispatches the traffic. This was used as feedback to the theoretical model resulting in higher delay values than originally anticipated.

As shown in chapter four, the bounds obtained by the theoretical model were not exceeded in the majority of the cases under consideration. Only in two cases the theoretical bound was exceeded by just a few microseconds. The theoretical model seems to be valid and these small deviations, observed in two out of eighteen cases examined, have to be further investigated. As it can be easily understood in such a complex system such small deviations can be attributed to a number of factors. Due to the specific time limitation of the thesis, these factors could not be further investigated.

Taking into consideration that 1Gbps Ethernet will become available for automotive and that future versions of AUTOSAR will support 802.1Q and 802.1p, great potential is being presented for the designers of future vehicular communication networks. Finally, it has to be mentioned that Ethernet can utilize both copper wire and optic fiber as physical medium. This provides additional flexibility for the design of the system.

The conclusions above were drawn from the research carried out within this thesis project and they can be considered as a starting point, regarding other future research work that may be carried out.

An important issue that has to be examined is that of the message overhead. The first and second priority messages had a payload of 32 bytes. This payload was encapsulated within the UDP header, followed by IP header and the Ethernet header adding a total overhead of 58 bytes. The valuable information travelling through the network is significantly lower than the overhead required for the transmission leading to waste of bandwidth. On the other hand, a large payload leads to higher delays. One possible solution is to pack a number of messages into one Ethernet frame. However, research has to be carried out to find the optimum solution regarding this trade-off.

A message may take an amount of time from the moment it is received from the network interface of an ECU to the moment it is delivered to its respective application. The delay calculated in this thesis was the end-to-end network delay. The application-to-application delay has to be further investigated. The specifications provided by Volvo Car Corporation required a 5ms deadline. The experimentally measured end-to-end delay values were significantly lower, indicating than the deadline of 5ms can be met. However, an ECU performs a lot of other tasks besides the ones needed for communication purposes. Therefore the processor utilization of the ECU is a factor that has to be taken into consideration. Having an IP stack with advanced functionality may increase the processor utilization.

As mentioned earlier, the only factor that may lead to packet loss is insufficient memory. Since the switches have a shared-memory architecture, lower priority traffic may fill the memory forcing the switch to drop incoming packets regardless of their priority. Therefore the calculation of the buffer size is of great importance. Network calculus enables the calculation of pessimistic backlog bounds. These bounds have to be experimentally evaluated so that design engineers will be aware of the exact amount of memory needed.

Finally for the needs of this thesis project strict priority policy was applied. A strict priority policy may lead lower priority queues to starvation. Therefore other priority policies have to be experimentally investigated. A policy suitable for automotive application may be the one described by Rahmani et al. in [22]. In this hybrid approach, strict priority policy is applied for the first priority data while for the following priorities a weighted fair queuing policy is applied.

References

[1] B. Muller-Rathgeber, M. Eichhorn, and H. U. Michel, "A unified Car-IT Communication-Architecture: Design guidelines and prototypical implementation," in *Intelligent Vehicles Symposium, 2008 IEEE*, 2008, pp. 709-714.

[2] L. Hyung-Taek, L. Volker, and D. Herrscher, "Challenges in a future IP/Ethernet-based in-car network for real-time applications," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, 2011, pp. 7-12.

[3] J. Rox, R. Ernst, and P. Giusto, "Using timing analysis for the design of future switched based Ethernet automotive networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, 2012, pp. 57-62.

[4] A. Kern, Z. Hongyan, T. Streichert, and J. Teich, "Testing switched Ethernet networks in automotive embedded systems," in *Industrial Embedded Systems (SIES), 2011 6th IEEE International Symposium on*, 2011, pp. 150-155.

[5] M. Rahmani, R. Steffen, K. Tappayuthpijarn, E. Steinbach, and G. Giordano, "Performance analysis of different network topologies for in-vehicle audio and video communication," in *Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008. IT-NEWS 2008. 4th International*, 2008, pp. 179-184.

[6] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer,* vol. 35, pp. 88-93, 2002.

[7] T. Nolte, "Share-driven scheduling of embedded networks," PhD Thesis, Computer Science and Electronics, Malardalen University, Sweden, 2006.

[8] (2013, 13 June). *MOST*. Available: http://www.mostnet.de

[9] P. Winzer, "Beyond 100G Ethernet," *Communications Magazine, IEEE,* vol. 48, pp. 26-30, 2010.

[10] (2013, 13 June). *Open Alliance*. Available: http://www.opensig.org/

[11] (2013, 13 June). *BroadR-Reach* Available: http://www.broadcom.com/products/features/connected_car.php

[12] (2013, 13 June). *RTPGE Study Group*. Available: http://www.ieee802.org/3/RTPGE/

[13] S. Furst, "Challenges in the design of automotive software," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, 2010, pp. 256-258.

[14] (2013, 13 June). *AUTOSAR*. Available: www.autosar.org

[15] D. Diekhoff, "AUTOSAR basic software for complex control units," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, 2010, pp. 263-266.

[16] B. Huang, H. Dong, D. Wang, and G. Zhao, "Basic Concepts on AUTOSAR Development," in *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*, 2010, pp. 871-873.

[17] R. L. Cruz, "A calculus for network delay. I. Network elements in isolation," *Information Theory, IEEE Transactions on,* vol. 37, pp. 114-131, 1991.

[18] R. L. Cruz, "A calculus for network delay. II. Network analysis," *Information Theory, IEEE Transactions on,* vol. 37, pp. 132-141, 1991.

[19] J.-Y. Le Boudec and P. Thiran, *Network calculus : a theory of deterministic queuing systems for the Internet*. New York: Springer, 2001.

[20] J. Jasperneite, P. Neumann, M. Theis, and K. Watson, "Deterministic real-time communication with switched Ethernet," in *Factory Communication Systems, 2002. 4th IEEE International Workshop on*, 2002, pp. 11-18.

[21] J.-Y. L. Boudec, "Network Calculus Made Easy," *Technical Report EPFL,* vol. 96, p. 30, 14 December 1996.

[22] J. P. Georges, E. Rondeau, and T. Divoux, "Evaluation of switched Ethernet in an industrial context by using the Network Calculus," in *Factory Communication Systems, 2002. 4th IEEE International Workshop on*, 2002, pp. 19-26.

[23] J. P. Georges, T. Divoux, and E. Rondeau, "Comparison of switched Ethernet architectures models," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 375-382 vol.1.

[24] J. P. Georges, T. Divoux, and E. Rondeau, "A formal method to guarantee a deterministic behaviour of switched Ethernet networks for time-critical applications," in *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, 2004, pp. 255-260.

[25] J. P. Georges, T. Divoux, and E. Rondeau, "Strict Priority versus Weighted Fair Queueing in Switched Ethernet Networks for Time Critical Applications," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, pp. 141-141.

[26] M. Rahmani, K. Tappayuthpijarn, B. Krebs, E. Steinbach, and R. Bogenberger, "Traffic Shaping for Resource-Efficient In-Vehicle Communication," *Industrial Informatics, IEEE Transactions on,* vol. 5, pp. 414-428, 2009.

[27] J. F. Kurose and K. W. Ross, *Computer networking a top-down approach featuring the internet*, 3d ed. Boston: Addison-Wesley, 2004.

[28] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched Ethernet," in *Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on*, 2004, pp. 13-22.

[29] (2013, 13 June). *ARCCORE*. Available: www.arccore.com

Appendix A

The switch fabric is the delay suffered by a frame passing through the switch without any queuing delays. In order to measure this delay, an experimental setup as the one shown in Figure 29 was deployed. Laptop A sends a ping message to Laptop B every 1ms. The nodes A to D exchange traffic between them in order to load the switch. For measuring the delay of a frame passing through the switch, the frame has to be time stamped while entering and exiting the switch. Then by subtracting the two time values the delay can be calculated. The first tap was connected on the link between Laptop A and Switch A, while the second tap was connected to the link between Switch A and Laptop B. The two taps were then connected to an experimental computer through the VN5610 network interface, which captured and time-stamped the frames. The trace of the captured traffic was analyzed using CANoe by Vector. The experiments were performed both for small sized frames (90 bytes) as well as maximum sized frames (1530 bytes). The worst case switch fabric delay value was found to be 10µs.

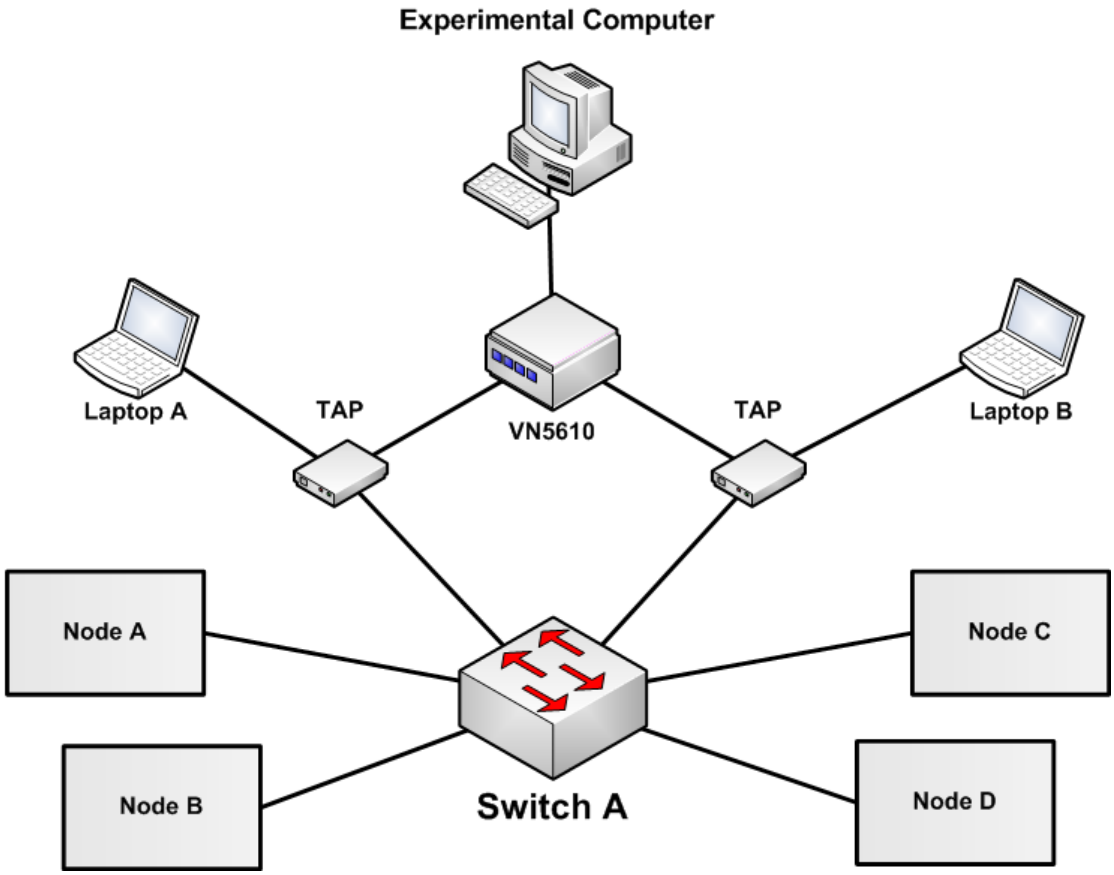


Figure 29. Switch Fabric measurement setup

Appendix B

```
/******** File: curve.java **********/

package netcalc;
import java.io.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;



public class curve {

    double LinkCapacity,MaxFrameLength,SmallFrameLength,AtoVCM,InputCapacity;
    double[] g,burst,rate,serviceRate,T;
    double[] T1;
    double[] T2;
    double[] Backlog1,Delay1,Backlog2,Delay2;
    double[] q,r;
    double fabric,SmallTransmission,MaxTransmission;

     public void init(){
        try {
            burst = new double[8];      //array for the burst size of the 4 flows on 2 switches
            rate = new double[8];       //array for the data rate of the 4 flows on 2 switches
            serviceRate = new double[4];//array for the service rate of the 4 flows
            T = new double[4];          //Delay caused by blocking and interference for every flow
            T1 = new double[4];         //T for round 1
            T2 = new double[4];         //T for round 2
            q = new double[4];          //new burst size after switch A
            r = new double[4];          //new rate after switch A
            g = new double[4];          //inflection point
            Backlog1 = new double[4];   //Backlog on switch A
            Delay1 = new double[4];     //Delay on switch A
            Backlog2 = new double[4];   //Backlog on switch B
            Delay2 = new double[4];     //Delay on switch B
            fabric=1E-5;                //switch fabric value


            //Open the text file to read input
            File file = new File("input.txt");
            Scanner in= new Scanner(file);
            in.useDelimiter(",");   //values are separated with ','
            Scanner sc;

            int i=-1;
            while(in.hasNextLine())
            {
                String str=in.nextLine();
                sc=new Scanner(str);
                sc.useDelimiter(",");
                if(i==-1){
                    //First line of input file has:
                    LinkCapacity=sc.nextInt();
                    MaxFrameLength=sc.nextInt();
                    SmallFrameLength=sc.nextInt();
                    SmallTransmission=SmallFrameLength/LinkCapacity;
                    MaxTransmission=MaxFrameLength/LinkCapacity;
```

```
            }
            else if(i<8){
                //the following lines give the arrival curves
                burst[i]=sc.nextInt();
                rate[i]=sc.nextInt();
            }
            else {
                AtoVCM=sc.nextInt();
            }
            i++;
          }


        } catch (FileNotFoundException ex) {
            Logger.getLogger(curve.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("No input file found!");
        }
    }

     //calculate the inflection point 'g' of flow 'i'
    void find_g(int i, int round) {

        if(round==1){
            InputCapacity = 3*LinkCapacity;
        }
        else if (round==2){
            InputCapacity = 2*LinkCapacity;
        }

        if(i<2){
            g[i]=burst[i]/(InputCapacity-rate[i]);
        }
        else{
            g[i]=burst[i]/(LinkCapacity-rate[i]);
        }
    }

    //calculate the delay value of 'T1' on flow 'i' for the first switch
    void find_T1(int i){
        if (i==0){  //First priority level
            T1[i]=2*MaxFrameLength/LinkCapacity;
        }
        else if (i==1){  //Second priority level
            T1[i]=burst[0]/(LinkCapacity-rate[0]) + 2*MaxFrameLength/LinkCapacity ;
        }
        else if(i==2){   //Third priority level
            T1[i]=+ (burst[0]+burst[1])/(LinkCapacity-rate[0]-rate[1]) +
2*MaxFrameLength/LinkCapacity;
        }
        else if(i==3){  //Fourth priority level
            T1[i]=(burst[0]+burst[1]+burst[2])/(LinkCapacity-rate[0]-rate[1]-rate[2]);
        }
    }

    //calculate the delay value of 'T2' on flow 'i' for the second switch
    void find_T2(int i){
        if (i==0){  //First priority level
            T2[i]= 2*SmallFrameLength/LinkCapacity ;
        }
        else if (i==1){  //Second priority level
```

```
        T2[i]= burst[0]/(LinkCapacity-rate[0]) ;
    }
    else{
        T2[i]=0;
    }
}

//calculate the service rate 'R' offered on flow 'i' for a 'round'
void find_R(int i, int round){
    if (i==0){
        serviceRate[i]=LinkCapacity;
    }
    else if (i==1){
       serviceRate[i]=LinkCapacity-rate[0];
    }
    else if(i==2){
        if(round==1)
        {serviceRate[i]=LinkCapacity-rate[0]-rate[1];}
        else if (round==2)
        {serviceRate[i]=LinkCapacity;}
    }
    else if(i==3){
        if(round==1)
        {serviceRate[i]=LinkCapacity-rate[0]-rate[1]-rate[2];}
        else if (round==2)
        {serviceRate[i]=LinkCapacity;}
    }
}

//calculates the backlog of flow 'i' on 'round'
void Backlog(int i,int round){

    if (round==1){

        if(T1[i]<=g[i]){
            Backlog1[i] = burst[i] + rate[i]*g[i] - serviceRate[i]*(g[i]-T1[i]);
        }
        else{
            Backlog1[i] = burst[i] + rate[i]*T1[i];
        }
    }
    else if(round==2){

       if(T2[i]<=g[i]){
            Backlog2[i] = burst[i] + rate[i]*g[i] - serviceRate[i]*(g[i]-T2[i]);
        }
        else{
            Backlog2[i] = burst[i] + rate[i]*T2[i];
        }

    }
}

//calculates the delay of flow 'i' on 'round'
void Delay(int i,int round){
    if (round==1){
        Delay1[i] = (burst[i]+rate[i]*g[i])/serviceRate[i] + T1[i]-g[i];
    }
    else if(round==2){
        Delay2[i] = (burst[i]+rate[i]*g[i])/serviceRate[i] + T2[i]-g[i];
```

```
    }

}

//put values of new burst size and rate in variables 'q' and 'r'
void set_q(int i){
    burst[i]=burst[i]+T1[i]*rate[i];
    q[i]=burst[i];
    r[i]=rate[i];
}

//update the 'burst' and 'rate' values for communication domain
void roundtwoVCM(){
    //burst and rate for flow1
    burst[0]=q[0]+burst[4];
    rate[0]=r[0]+rate[4];

    //burst and rate for flow2
    burst[1]=q[1]+burst[5];
    rate[1]=r[1]+rate[5];
}

//update the 'burst' and 'rate' values for infotainment domain
void roundtwoDIM(){

    //burst and rate for flow1
    burst[0]=q[0]+burst[6];
    rate[0]=r[0]+rate[6];

    //burst and rate for flow2
    burst[1]=burst[7];
    rate[1]=rate[7];
}

//calculate delays and backlogs for switch A
public void first_calc(){

    for (int i=0; i<4; i++ )
    {
        find_g(i,1);    //find inflection point
        find_T1(i);     //find delay parameter T
        find_R(i,1);    //find service rate of the switch
        Backlog(i,1);   //calculate the backlog
        Delay(i,1);     //calculate the delay
        set_q(i);       //restore burst and rate values
                        //for next round calculations
    }
}

//calculate delays and backlogs for switch 2
public void second_calc(){

    for (int i=0; i<4; i++ )
    {
        find_g(i,2);
        find_T2(i);
        find_R(i,2);
        Backlog(i,2);
        Delay(i,2);
    }
```

```
   }

   //print results
   public void results(){


System.out.println("=============================================================
============================");
       System.out.println("Flow \tBuffer1 \tDelay1 \t\tBuffer2 \tDelay2 \t\tTotalDelay");
       for(int i=0;i<4;i++){
          System.out.format("%d",i+1);
          System.out.format("\t%d",(int)Backlog1[i]);
          System.out.format("\t\t%d",(int)(Delay1[i]*1000000));
          System.out.format("\t\t%d",(int)Backlog2[i]);
          System.out.format("\t\t%d",(int)(Delay2[i]*1000000));

          SmallTransmission=(SmallFrameLength-12)/LinkCapacity;
          MaxTransmission=(MaxFrameLength-12)/LinkCapacity;

          if(i<2){

System.out.format("\t\t%d%n",(int)((Delay1[i]+Delay2[i]+2*SmallTransmission+2*fabric)*1000
000));
          }
          else{

System.out.format("\t\t%d%n",(int)((Delay1[i]+Delay2[i]+2*MaxTransmission+2*fabric)*10000
00));
          }
       }

System.out.println("=============================================================
============================");
      int buffer=0;
      for(int i=0;i<4;i++){
         buffer+=Backlog1[i];
      }
      System.out.format("Total backlog %d => %dKB%n", buffer, buffer/1024);

System.out.println("=============================================================
============================");
   }

}
/******** End of File: curve.java **********/

/******** File: Netcalc.java **********/

package netcalc;

public class Netcalc {

   public static void main(String[] args) {

       curve traffic= new curve(); //create a new class curve
       traffic.init();             //initialize the variables


////////////////////////////////////////////////////////////////////////
//       calculate delays and backlog on first switch
```

```
////////////////////////////////////////////////////////////////////////

        traffic.first_calc();

////////////////////////////////////////////////////////////////////////
// calculate delays and backlog on second switch for Communication domain
////////////////////////////////////////////////////////////////////////

        traffic.roundtwoVCM();  //update the variables
        traffic.second_calc();  //perform the calculations
        traffic.results();      //print the results

////////////////////////////////////////////////////////////////////////
// calculate delays and backlog on second switch for infotainment domain
////////////////////////////////////////////////////////////////////////

        traffic.roundtwoDIM();
        traffic.second_calc();
        traffic.results();
    }
}

/******** End of File: Netcalc.java **********/
```

Appendix C

For an MTU of 1500 bytes

- Use Case 2

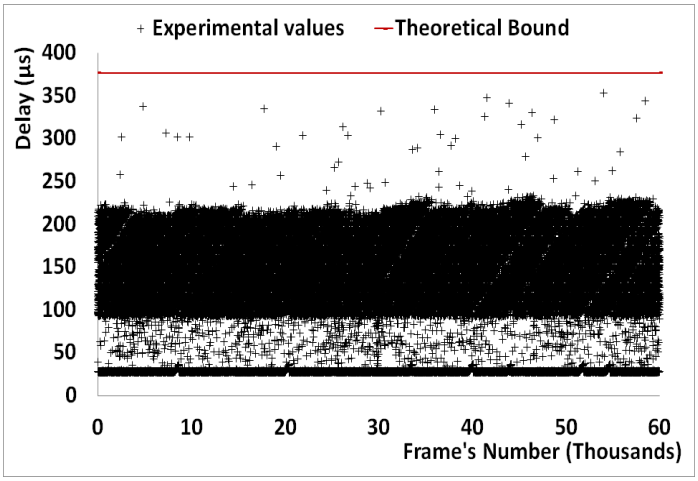


Figure 30. Case 2 second priority delay values

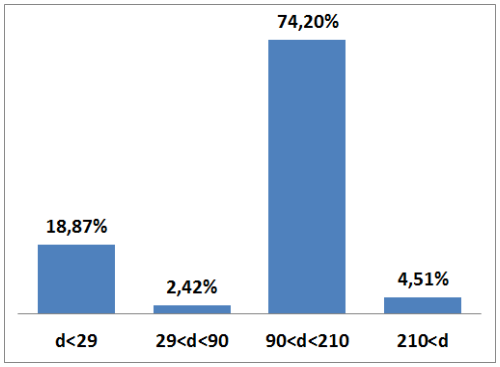


Figure 31. Case 2 second priority delay spread

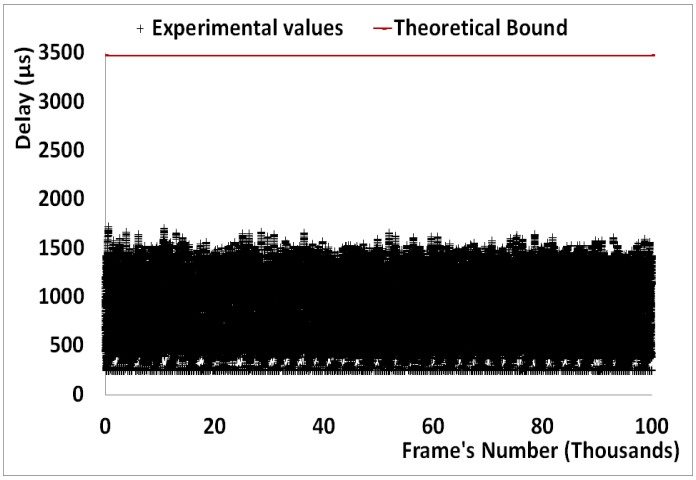


Figure 32. Case 2 third priority delay values

- Use Case 3

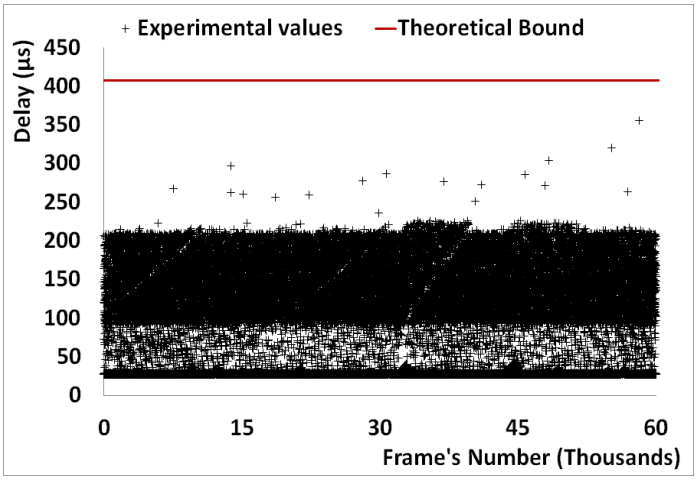


Figure 33. Case 3 second priority delay values

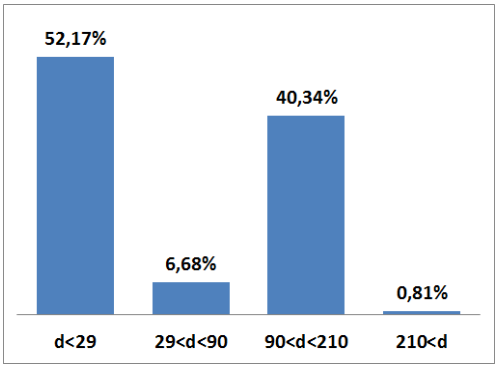


Figure 34. Case 3 second priority delay spread

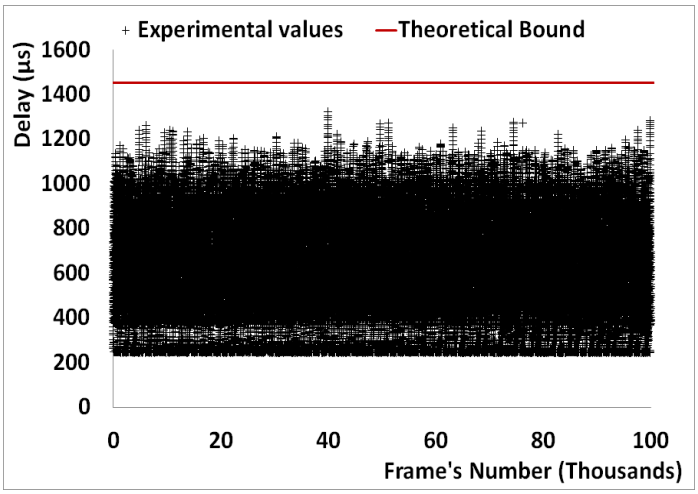


Figure 35. Case 3 third priority delay values

For an MTU of 800 bytes

- Use Case 1

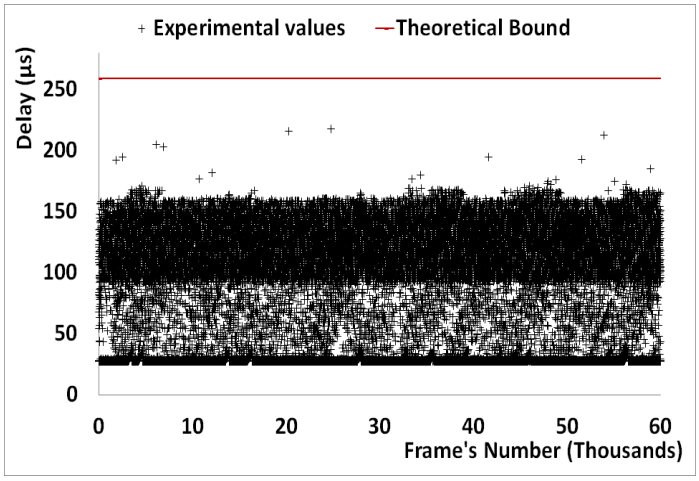


Figure 36.Case 1 second priority delay values (MTU 800)

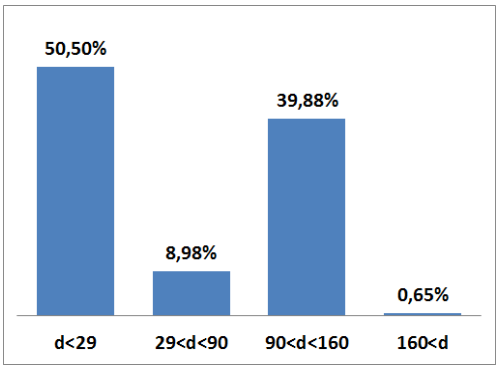


Figure 37. Case 1 second priority delay spread

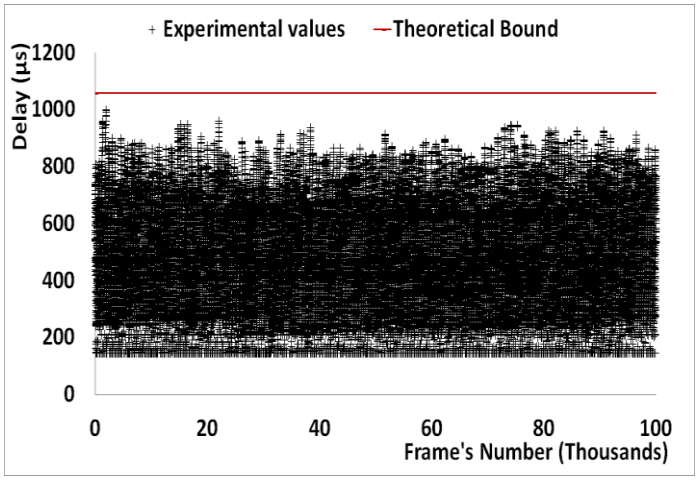


Figure 38. Case 1 third priority delay values (MTU 800)

- Use Case 2

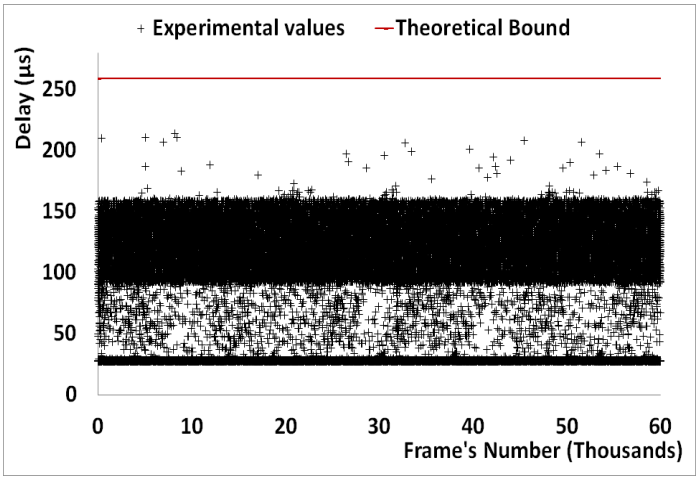


Figure 39. Case 2 second priority delay values (MTU 800)

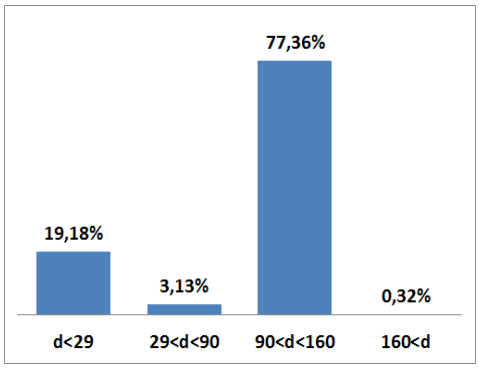


Figure 40. Case 2 second priority delay spread

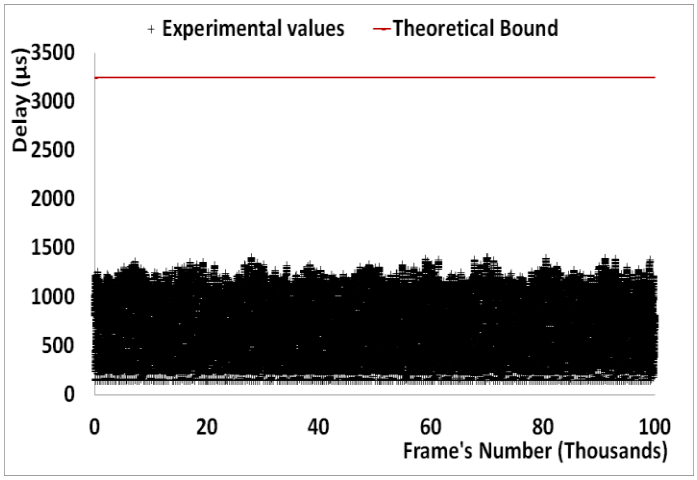


Figure 41. Case 2 third priority delay values (MTU 800)

- Use Case 3

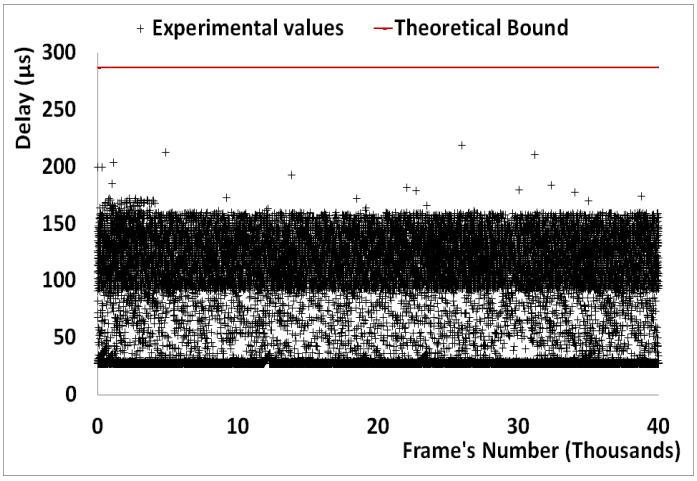


Figure 42. Case 3 second priority delay values (MTU 800)

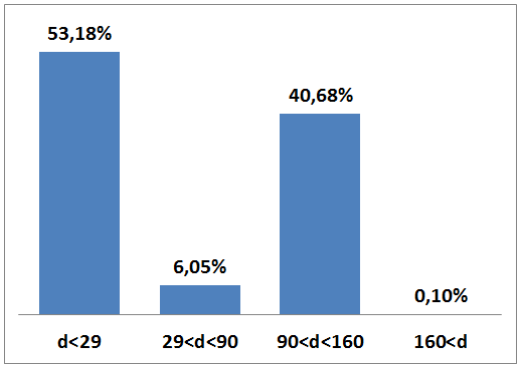


Figure 43. Case 3 second priority delay spread

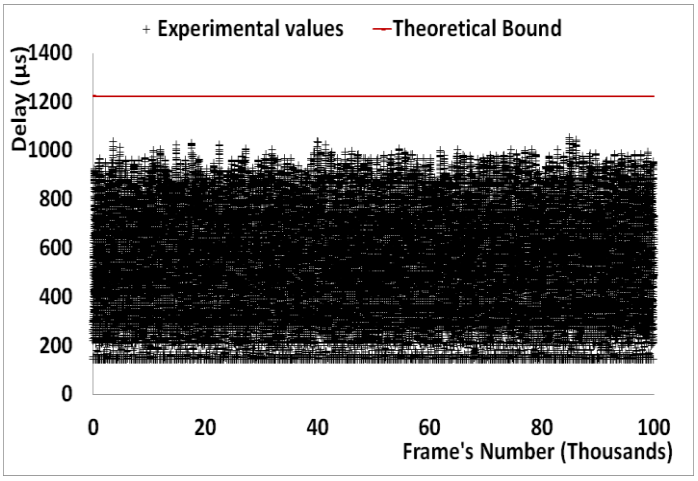


Figure 44. Case 3 third priority delay values