DEGREE PROJECT IN COMMUNICATION SYSTEMS, FIRST LEVEL
STOCKHOLM, SWEDEN 2014

Automobile Control Systems

Transition from Controller Area Networks to Ethernets

RASMUS EKMAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Automobile Control Systems

Transition from Controller Area Networks to Ethernets

Rasmus Ekman

2014-07-19

Kandidatexamensarbete

Examiner and academic adviser
Professor Gerald Q. Maguire Jr.

School of Information and Communication Technology (ICT)
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

Due to concerns about the negative impacts of powering vehicles using fossil fuel and the future availability of fossil fuel, there has been an increased focus on electric vehicles. However, current electric vehicle energy efficiency is a key problem as these vehicles are not as efficient as fossil-fueled vehicles. One way of decreasing a vehicle’s energy consumption is to reduce the weight of the vehicle, while still ensuring the safety and reliability of the vehicle. Controller Area Network (CAN) systems have been used in vehicles to realize real-time applications, however the low peak data rates of CAN have begun to limit the applications that can be realized.

This bachelor’s thesis project focuses on secure communication *within* a vehicle using Ethernet. Additionally, the use of Power over Ethernet can be used for powering some of the network attached devices within the vehicle. The goal is to reduce the number of components and the weight of the vehicle while continuing to ensure the security and reliability of the communication – even when the network grows in size (either in physical size or in number of connected devices).

This thesis shows that an Ethernet based system can serve as a possible replacement candidate for the CAN system due to its low latencies and high bandwidth. Ethernet is also a very scalable system with none of the limitations that a CAN system have.

Keywords: Ethernet based communcation, Power over Ethernet

Sammanfattning

Den negativa påverkan av fossila bränslen har de senaste årtionden haft en negativ på planeten, mängden fossila bränslen över världen konsumeras även i en högre takt än vad som produceras. Därför har fokusen för att finna förnybara energi källor som både är effektiva och inte påverkar miljön på ett negativt sätt ökat. Därför är elbilar en viktig del i konverteringen av enheter som drivs av fossila bränslen till förnybara energikällor.

Ett av problemen i en elbil är att energi konsumptionen är inte lika effektiv som fossila bränslen inom bil industrin. Ett sätt att sänka energi konsumptionen är att minska mängden komponenter inom en bil för att minska på vikten, utan att påverka säkerheten och tillförlitligheten. Tidigare har man använt sig av ett CAN system för att försäkra sig om systemet fungerar felfritt i realtid, problematiken med detta system är att när nätverket ökar i storlek så sätter de fysiska begränsningarna av detta system stop för den garanterade säkerheten.

Detta kandidatexamensarbete kommer att fokusera på den interna kommunikationen i en elbil med hjälp av ett ethernet baserat kommunkations system över CAN systemet. Power over Ethernet tekinken kommer att tillämpas för de systemen som kan drivas av detta system. Målet är att reducera antalet komponenter som behövs och att garantera säkerheten och tillförlitligheten av den interna kommunikationen när nätverket av komponenter ökar i storlek.

Det här kandidatarbetet visar att Ethernet kan ersätta det nuvarande CAN systemet eftersom att Ethernet erbjuder låga latenser och hög bandbredd. Detta arbete visar även att Ethernet är väldigt skalbart och har inte begränsingarna som ett CAN system har.

Nyckelord: Ethernet baserad kommunikation, Power over Ethernet

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

ACK	Acknowledge
ASCII	American Standard Code for Information Interchange
CAN	Controller Area Network
COTS	Commercial-Off-The-Shelf
CSMA/CD	Carrier Sense Multiple Access/Collison Detection
DOS	Denial-of-service
DNS	Domain Name System
EMI	Electromagnetic interference
EBCDIC	Extended Binary Coded Decimal Interchange Code
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv4	IP version 4
IPv6	IP version 6
IPsec	IP security
ISP	Internet Service Provider
LAN	Local Area Network
MAC	Media Access and Control
Mbps	Megabit per second
MTU	Maximum Transmission Unit
OSI model	Open Systems Interconnection Model
PoE	Power over Ethernet
RBS	Raspberry pi
RTT	Round-trip time
SCTP	stream control transmission protocol
TCP	Transmission Control Protocol
UDP	User datagram protocol

1 Introduction

This chapter gives the reader an understanding of the limitations of Controller Area Network (CAN) systems and the strengths of Ethernet. The goal is to lay a foundation for how these techniques that can be used in combination with Ethernet to provide a reliable error detection system capable of replacing CAN in error sensitive systems. The physical limitations of CAN and the properties of Ethernet technologies as realized today motivate the consideration of Ethernet as a viable replacement for CAN.

1.1 Problem definition

Today CAN bus based communication is widely used in vehicles. However, in order to ensure successful data transfer between end-points in the network there are a number of constraints on this bus. One of the limitations of CAN bus is that the transfer of data fragments is limited to a maximum data rate of 1 Megabit/s (Mbps). In order to achieve this data rate *without* affecting the other devices on the bus the physical wire length between devices has to be limited. If the wire length is increased, then the data rate must be decreased to ensure the correctness and reliability of the communication. Additionally, there a limitation on how many nodes can be connected to the network. In a high speed-CAN system, the maximum number of nodes is 110 with a maximum total wire length of 6500 meters [1], [2].

The maximum data rate of CAN bus limits it applications in modern vehicles to relatively low data rate applications. Additionally, CAN bus attached devices also need to be connected to other wires to provide power to the device. In contrast, Ethernet devices can support 10, 100, and 1000 Mbps while also providing a limited amount of power using Power over Ethernet (PoE). Maximum link distances at these data rates are ~100 m with CAT 6 cable (of which 90 m is assumed to be solid core wiring, rather than stranded cable).

1.2 Goals

The goal of this thesis project is to show that an Ethernet based communication system can serve as a potential alternative to the well-established CAN bus systems. Subgoals include showing that the error detection in a CAN system can be provided by the Ethernet based system and showing that the Ethernet based network can be scaled up to an appropriate size *without* suffering from the same problems that a CAN system does. Additionally, show that PoE can be used to power a variety of network attached devices that might be attached to network in a modern electrical vehicle (this includes consideration of what limitations there might be on such devices' power requirements).

1.3 Limitations

All of the measurements reported in this thesis were made using consumer grade equipment. The details of this equipment are in Section 3.1.

1.4 Structure of the thesis

Chapter 1 gave the reader an understanding of the general problem addressed in this thesis. Chapter 2 gives a brief and basic explanation of the different underlying technologies that are relevant to the use of an Ethernet based communication system within an electrically powered vehicle. Chapter 3 describes the method that will be used to reach the goal stated in Section 1.2. Chapter 4 describes the measurements and methods used to show that an Ethernet based system can be used as a replacement for CAN based system. Chapter 5 describes the analysis of the data collected. Chapter 6 concludes the thesis with some conclusions, suggestions for future work, and with some reflections on this thesis from social, economic, and sustainability points of view.

2 Background

This chapter provides a brief description of an embedded platform and some concepts that are useful to understand the rest of this thesis.

2.1 Open Systems Interconnection (OSI)-model

The Open Systems Interconnection (OSI) model was defined as a standardized abstraction of network protocols. Each protocol, be it implemented in hardware or software, can be placed within a *layer*. In order to make a working network all of the relevant layers are combined to form a protocol stack. A protocol stack uses the protocol in each layer to transfer the data between two end-points at the corresponding layers. When using a protocol from the application layer (see Section 2.1.1) the data propagates through the OSI model as shown in Figure 2-1, down to the physical layer where the data is transferred to another physical layer end-point in the network. Once the data reaches the other physical layer end-point the data propagates up from the physical layer up to the application layer.

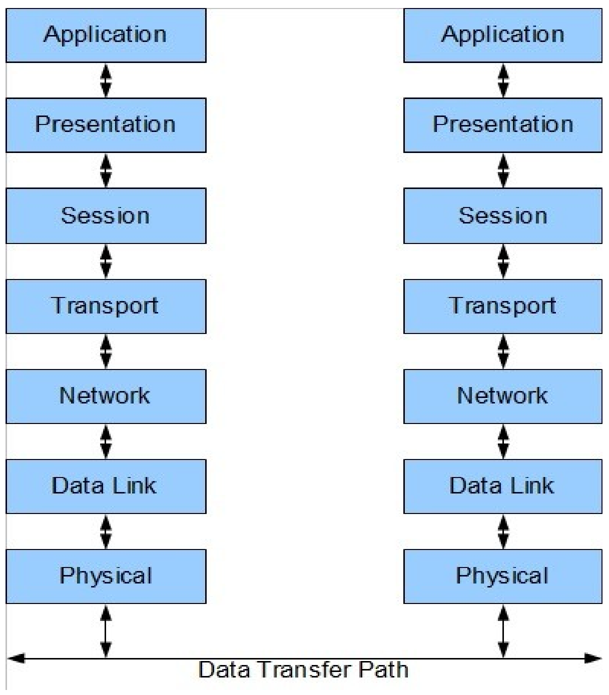


Figure 2-1: OSI model

2.1.1 Application Layers

Software functions reside in the application layer. This layer includes protocols such as Hypertext Transfer Protocol (HTTP), Domain Name System (DNS) and File Transfer Protocol (FTP). The application layer also realizes Application Programming Interfaces (APIs) that developers use to communicate between applications over a network.

2.1.2 Presentation Layer

When transferring data across a network each system receiving the data can interpret the data differently. The presentation layers solves this by formatting the information so that the application layer does not have to interpret the data. For example, the presentation layer could convert text from Extended Binary Coded Decimal Interchange Code (EBCDIC) to an American Standard Code for Information Interchange (ASCII).

2.1.3 Session Layer

The session layer includes mechanism to open, manage and close sessions.

2.1.4 Transport Layer

The transportation layer provides a host-to-host service, in this layer protocols can provide a reliable data transfer protocol or a non-reliable data transfer protocol. Examples of these protocols are User Datagram Protocol (UDP, see Section 2.3.2), Transmission Control Protocol (TCP, see Section 2.3.3), and Stream Control Transmission Protocol (SCTP, see Section 2.3.4).

2.1.5 Network Layer

The network layer is responsible for forwarding packets so that each packet reaches the end host. This layer is a connectionless communication between hosts, meaning the destination host does not need to acknowledge the packets they receive, the acknowledge is done in the transportation layer protocols. Two of the protocols in this layer is the Internet Protocol (IP, see Section 2.3) and Internet Control Message Protocol(ICMP, see Section 2.3.1). The network layer also give each host in a network a unique address, when this address is used in reference to the Internet this address is called an *IP address.*

2.1.6 Data Link Layer

In the data link layer the protocol transfers data to adapters in both Local Area Network (LAN) and in wide area networks and also offers support for error-checking in case the data was altered or corrupted during the transfer in the physical layer. One of the technologies for this layer is Ethernet.

2.1.7 Physical Layer

The physical layer is where the data is transferred with the use of electrical or optical signals depending on what communication medium that is used in the hardware. Technologies in this layer are the CAN and different versions of Ethernet.

2.2 Ethernet

Ethernet is the most widely used technology in wired LANs because it is easy for network administrators to deploy & manage it and Ethernet cabling & interfaces are low cost due to the wide variety of cost effective Commercial-Off-The-Shelf (COTS) products. Ethernet is continuously evolving to provide increasing performance and stability, both in terms of data rates and wiring. In the IEEE 802 series of standards for IEEE 802.3 the standardized MTU size is 1500 bytes, but by using jumbo frames this can be increased, see Section 2.5.1 for further information regarding jumbo frames.

Ethernet is a CSMA/CD based communication system, but with the exception that it shares the medium with another link only. And if both endpoint's of the link are using full-duplex then there is no need for any of the endpoint's to check if the medium is being used. This is because that communication is bi-directional and therefor the endpoint's can send data over the cable freely.

2.2.1 Ethernet Frame

Figure 2-2 shows the format of an Ethernet frame[1]. An explanation of each of these fields is given below.

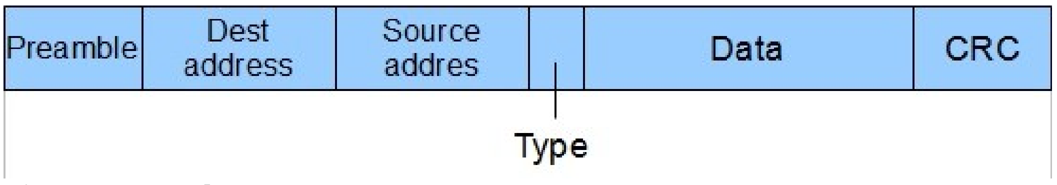


Figure 2-2: Ethernet Frame

[1] The official standard not considering Jumbo Frames (See section 2.5.1).

Preamble (8 octets)	The preamble enables the destination interface to synchronize its clock with the transmitting interface.
Destination Address (6 octets)	The destination address contains the MAC address of the adapter that the frame is destined to.
Source Address (6 octets)	The source address contains the MAC address to the adapter that originated the frame.
Type Field (2 octets)	This field indicates the type of the frame.
Data Field (46-5000 octets)*	This field contains the IP packet or other data that is being transmitted.
CRC (4 octet)	The CRC field enables the receiver to check for bit errors in the frame.

2.2.2 Ethernet Switch Latency

One of the most common ways of interconnecting devices with Ethernet interfaces is to connect each of the devices to an Ethernet switch. Typically, such a switch will operate at the logical link layer. However, today many switches also have IP routing, firewall, network address translation, and other functions integrated into them.

2.2.2.1 Cut-Through Switch

One type of Ethernet switches is a cut-through switch. This type of switch quickly forwards frames that the switch receives. This type of switches has low latency because the switch has been constructed to start forwarding the frame as soon as the destination of the Ethernet frame has been determined. That means that once the 14 first bytes have been analyzed by the switch the frame is forwarded, i.e., as the frame is arriving it is being forwarded. The downside of this that since the CRC byte sequence in the end of the Ethernet frame, the CRC of the frame cannot be calculated and compared to the CRC in the frame before forwarding the frame. Therefore, if the forwarded frame contains any corrupted bits it will still be forwarded. A cut-through switch leaves the error-checking of frames to other network devices. If a network uses only cut-through switches, then error-checking and handling is done at the destination.

An alternative version of a cut-through switch is called a fragment free cut-through switch. This version buffers the frame until sufficient many bytes have arrived to ensure that there was no collision on the source port. Since this feature adds latency to the network this feature should only be used if collisions can be a problem. Note that in most switched Ethernets there are no collisions.

Cut-through switches are an important type of switches when it comes to realizing a network with a latency below 10 milliseconds (ms). Such low latency is often a requirement of a high-performance computing application [3].

2.2.2.2 Store-and-Forward Switch

A store-and-forward switch has higher latency than a cut-through switch. This higher latency is due to the fact that a store-and-forward stores the incoming frame until its fully received and the CRC has been checked to see if there were errors in this frame. If the switch detects an error, then this frame is dropped.

2.2.2.3 Last In First Out (LIFO)

A LIFO based queuing system can potentially deliver a low-latency system, but it does not offer a fair queuing system for applications that use the network. The delay within a Ethernet switch with a LIFO queue can be calculated by starting a timer t0 when the entire frame has been recieved and stopping it when the first bit leaves the Ethernet switch at t1 [4]. Figure 2-3 illustrates the latency of such a LIFO queue.

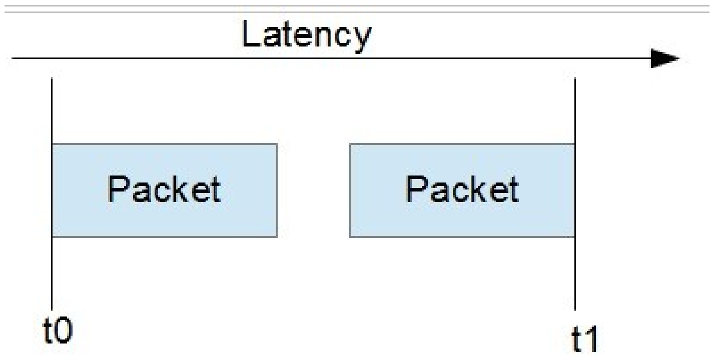


Figure 2-3: LIFO Latency

2.2.2.4 Last In Last Out (LILO)

The LILO queue system is a fair queuing system for applications. However, if the system is congested the latency of the network will increase since that it takes longer for each frame to be delivered to its destination as the frames have to wait their turn in one or more queues. Latency can be calculated by starting a timer at t0 when the entire frame has been recived and stopping the timer at t1 once the entire frame has been transmitted [4]. Figure 2-4 illustrates the latency of a LILO queue.

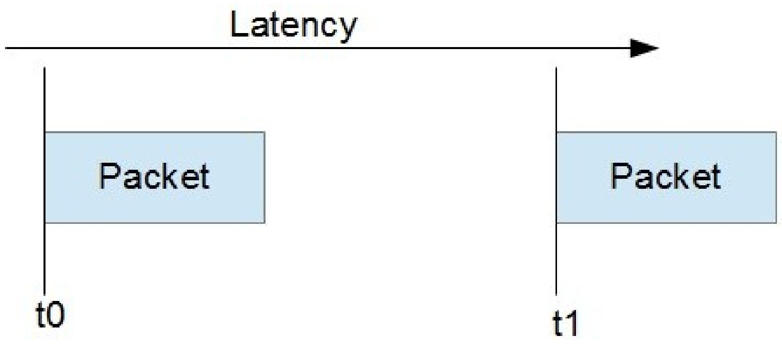


Figure 2-4: LILO Latency

2.2.2.5 First In First Out (FIFO)

A FIFO queuing system works in the same way as a LILO system and they perform equally due to the common features in how the systems work. To calculate the latency of a FIFO queue the t0 timer starts when the first bit is detected on the source port and t1 is when the first bit leaves the switch (see Figure 2-5).

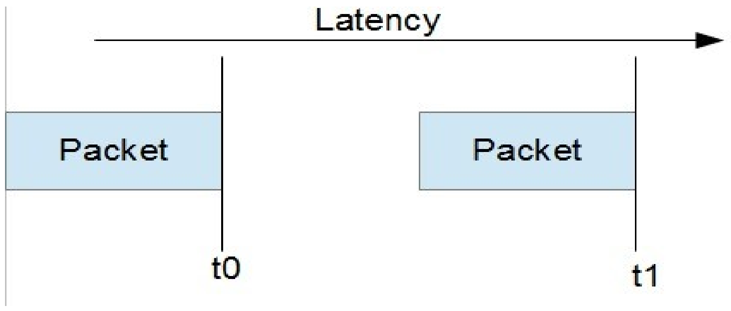


Figure 2-5: FIFO Latency

2.2.2.6 First In Last Out (FILO)

A FILO queuing system is shown in Figure 2-6. In this case the latency is computed from when the first bit is received until the last bit is forwarded.

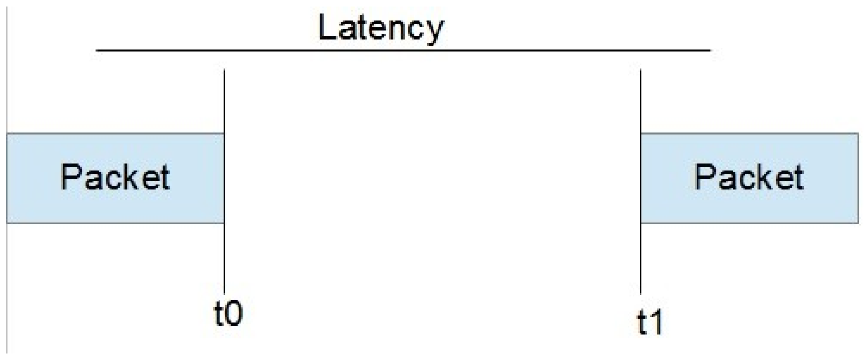


Figure 2-6: FILO Latency

2.3 Internet Protocol and transport layer protocols

The internet protocol (IP) is the network layer protocol in the TCP/IP protocol stack. The internet control message protocol (ICMP) is used within the network layer for management purposes. While at the transport layer, the two most widely used protocols are UDP and TCP. An additional transport protocol that is becoming increasingly popular is SCTP. Each of these protocols is briefly described in the following subsections.

It is important to note that today there is a transition being made from IP version 4 (IPv4) to IP version 6 (IPv6). Details about these two different versions can be found in any textbook on computer communications, such as [5]–[7]. In this thesis project, we will focus on IPv6 as this enables the largest address space and features auto configuration, cryptographic address generation, and supports IP security (IPsec).

2.3.1 ICMP

ICMP is used by network layer entities to communicate with each other. For example, ICMP can be used to inform a source that a datagram could not reach its destination [6], [8]. ICMP supports Router Solicitation and Advertisement – so that hosts can learn about routers and so that routers can advertise their availability.

2.3.2 UDP

UDP is a connectionless protocol for sending datagrams, i.e., it does not require any connection to be established before it initiates communication with a destination. UDP provides no functions for detecting missing or duplicated datagrams. When a destination receives a UDP datagram if the UDP checksum field is non-zero, then it computes a checksum over the datagram and a pseudo header that includes the source and destination IP addresses and protocol, to check that it has received this datagram from the indicated source IP address and that the datagram was received successfully. However, if it detects an error it does not inform the source host about their being an error, but rather the receiver silently discards the datagram. The UDP header is smaller than the TCP header. The low complexity of UDP allows UDP to have a lower latency than TCP [9].

2.3.3 TCP

TCP is a reliable byte stream transfer protocol with the ability to detect missing, damaged, or duplicated segments that are being transferred between two hosts. This communication begins with both hosts opening up a connection between them, thus the destination host knows that some data will be transferred to it and where this data comes from. To ensure that the source host knows that the destination host has retrieved all of the bytes up to some point it waits for acknowledgments (ACKs) from the destination host indicating the next expected byte's sequence number. If this ACK is not received by the source host within the expected time period, then the source host assumes that the data was not correctly delivered and retransmits the unacknowledged data. Duplicate data is silently ignored.[10], [11]

Since that TCP offers reliable delivery it must ensure that the data that was sent has not been corrupted or altered on the way by computing a checksum over the TCP segment and placing this checksum in the TCP header. If the destination's computation of the checksum does not match the checksum in the header, then the segment is silently discarded. The TCP protocol handles corrupted or lost segments based upon timeouts. Following the timeout the unacknowledged data will be retransmitted by the source host. When the TCP source receives three duplicate ACKs, then it knows that subsequent segments are being received so there is no congestion, but rather a loss has occurred, thus rather than waiting for a timeout it can immediately retransmit the unacknowledged data [5], [12].

2.3.4 SCTP

Stream Control Transmission Protocol (SCTP) is another transport protocol. Many applications have utilized TCP and UDP to transfer data between end-points, but as timing and reliability have become more important these two protocols work proved inadequate to meet some applications' requirements. SCTP provides the reliable transfer from TCP without the delay problems TCP's order-of-transmission causes. Unlike TCP and UDP, SCTP also provides multi-homing. Multi-homing allows system designers to easily manage and exploit link and path-level redundancies [10], [11].

Mutli-homing is possible in SCTP because the addressing of a SCTP endpoint can include a list of IP address, see Figure 2-7. The endpoints in Figure 2-7 only have one network interface so the SCTP address for the endpoints will be [25.11.124.166:100] and [213.112.62.123:200] for the interfaces on the left and right (respectively). Several IP addresses can be associated with one SCTP port. If the endpoints in Figure 2-7 were merged and have the same port for example port 100, then the SCTP address for that port would be represented by (A). If the endpoints in Figure 2-7 were merged and had different applications running on different interfaces the SCTP endpoint address would be presented as in (B) [11]. Multiple interfaces can be useful for load balancing or providing fail-over.

endpoint = [25.11.124.166, 213.112.62.123:100] (A)

endpoint = [25.11.124.166:100, 213.112.62.123:200] (B)

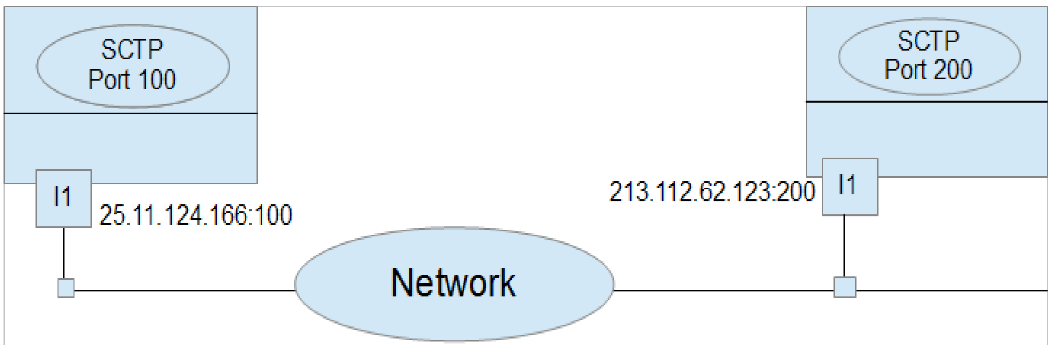


Figure 2-7: SCTP end-point address

When a computer is providing a service, it can be a potential target of DOS attacks. A SYN flood is one of such attack. This type of attack exploits the TCP handshake procedure to fool the system into using up all the system's resources on fake connections. Unfortunately, it is difficult to differentiate malicious TCP SYN-packets from non-malicious TCP SYN-packets because the source IP address is usually spoofed in this sort of attack. However, the SCTP protocol avoids this problem due to its four-way handshake procedure, as shown in Figure 2-8. When a client wants to initiate a connection, called an association in SCTP, the client sends a INIT request to the server. When the server receives the INIT request the server creates a cookie with the system resources in it, it does not allocate the resources needed. This prevents a SYN-flood DOS attack since no resources are actually allocated. The server send a INIT-ACK to the source address of the INIT packet. If the source address was spoofed, then there will be no association established between the endpoints. If the INIT request was from a non-malicious endpoint then this end-point will send back a COOKIE-ECHO packet to ensure the server that this a non-malicious association. The server will respond with a COOKIE-ACK packet which will confirm the association [11].

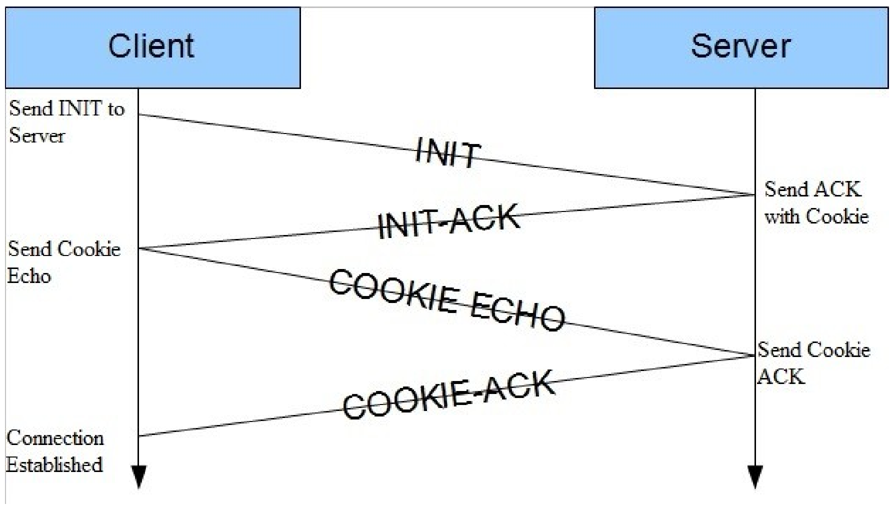


Figure 2-8: SCTP handshake procedure

The SCTP packet format and common header are shown in Figure 2-9. The fields of the packet are described below.

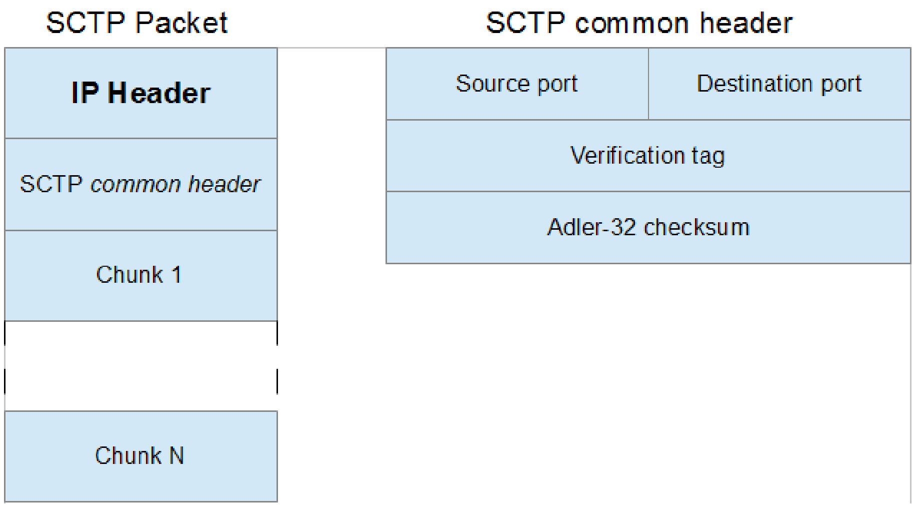


Figure 2-9: SCTP packet and common header structure

SCTP Packet	
Chunk	Contains either control information(control chunks) used to maintain the SCTP association or user data(data chunks). If both control chunks and data chunks exists then control chunks will always be the first chunks in the SCTP packet.
SCTP common header	
Source Port	Contains the sender endpoint's SCTP port.
Destination Port	Contains the receiving endpoint's SCTP port.
Verification tag	In the case where an association is terminated and a new one is established this tag verifies if the packets belong to the new association or the old one.
Adler-32 checksum	This checksum is used to compute if any part of the SCTP packet was altered or corrupted during transfer. The checksum covers the SCTP common header and all of the chunks in the package.

2.3.5 SCTP, TCP, and UDP

Table 2-1 shows a comparison of the three main IP transport protocols used today. We can see from this comparison that SCTP combines the features of TCP and UDP.

Table 2-1: Comparison of three transport protocols

Protocol Feature	SCTP	TCP	UDP
State required at each endpoint	Yes	Yes	No
Reliable data transfer	Yes	Yes	No
Congestion control and avoidance	Yes	Yes	No
Message boundary conversation	Yes	No	Yes
Path MTU discovery and message fragmentation	Yes	Yes	No
Multi-homed hosts support	Yes	No	No
Multi-stream support	Yes	No	No
Unordered data delivery	Yes	No	Yes
Security cookie against SYN flood attack	Yes	No	No
Built-in heartbeat(reachability check)	Yes	No	No

2.4 Controller Area Network(CAN)

The CAN bus system was developed by Robert Bosch GmbH to ensure a reliable data transfer between real-time peripheral devices, sensors, actuators, and controllers. This system is not affected by changes to the network, thus if a system such as headlights is added, removed, or non-operational this does not affect any other devices on the bus. The system allows multiple masters on the bus; this means that data transfer can be done in both directions. Use of CAN bus reduces the amount of wiring in the vehicle because the bus is shared between all devices connected to the bus. The system can detect errors and handles errors by retransmitting the corrupted data over the bus. It is also capable of differentiating between temporary and permanent failures of specific nodes, and has the possibility to switch off defective nodes [13]. CAN operates in the two lowest layers of the ISO OSI model. The CAN frame format is shown in Figure 2-10. The fields of this frame are described below.

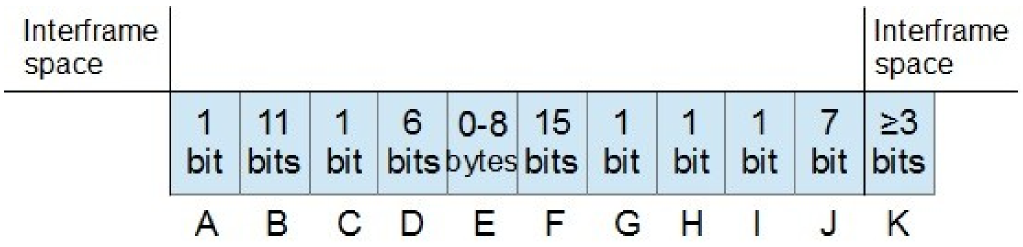


Figure 2-10: CAN frame structure

A	*Start of Frame*	Indicates the beginning of a message on the bus.
B	*Identifier*	Identifies the message and the priority of the message.
C	*RTR bit*	The remote transmission request(RTR) indicates if the frame is a data frame or a remote frame.
D	*Control*	Two bits reserved for future, DLC0-3 is data length code.
E	*Data Field*	Contains the data to be transferred over the bus.
F	*CRC Sequence*	These bits are used to check for errors.

G	*CRC Delimiter*	Used to separate the CRC bits from the ACK bit.
H	*ACK Slot*	Sends an acknowledgment bit.
I	*ACK Delimiter*	Used to separate the ACK bit from the End of Frame bits.
J	*End of Frame*	Indicates the end of the frame.
K	*Interframe*	In order for a node to transmit a message over the bus it must wait at least 3 bit times to ensure the bus is potentially free to use.

CAN bus utilizes a Carrier Sense Multiple Access/Collision Detection with Collision Detection (CSMA/CD) media access and control (MAC) protocol. This MAC protocol ensures that only one device at the time can access the bus by sniffing the bus to see if there is any data transfer in progress, if no data transfer is in progress, then the host can send data over the bus. Additionally, this bus supports Arbitration on Message Priority, so that higher priority messages get access to the bus before lower priority messages.

The use of carrier detection limits the length of the bus as all attached nodes must be able to detect that a transmission is in progress before they attempt to send any data. Additionally, if the bus is in use then both the MAC protocol and the priority mechanism can increase the delay before a message can be transmitted; hence the system might not be able to meet real-time guarantees [1]. The delay for a node to send a message can also be affected by quality of components and length of the bus, for a typical CAN system that follows the CAN 2.0b specification the speeds have limitations, for some general speeds see the table 2.1.

Table 2-2: Bit rate relative to length (Data from [21])

Bit rate	Bit time(μs)	Bus length(m)
1 Mb/s	1	30
800 kb/s	1.25	50
500 kb/s	2	100
250 kb/s	4	250
125 kb/s	8	500
62.5 kb/s	16	1000
20 kb/s	50	2500
10 kb/s	10	5000

2.5 What have others already done?

2.5.1 Jumbo frames

Jumbo frames are Ethernet frames with a size of up to 5000 bytes, rather than the conventional frame size of 1500 bytes. Arjun Reddy Kanthla shows that when using jumbo frames in a virtual machine in a loss-less laboratory environment jumbo frames could improve network performance in both virtual and physical environments [14].

Shaneel Narayan and Paula Raymond Lutui also showed that using jumbo frames rather than normal sized frames in a laboratory setup increased throughput. They also showed that using IPv6 over IPv4 the significantly decreased delay [15].

2.5.2 Ethernet and CAN comparison

Konrad Etschberger did some preliminary calculations regarding how well Ethernet compares to CAN when sending tiny messages, i.e., payloads of only a few bytes. This was done to see what the theoretical maximum frames rate would be (in frames per second (FPS)). When making these calculations he assumed an effectiveness of 20% in an Ethernet based communication system to avoid any collisions that could change the meaning of the messages. In a switched Ethernet network however there are no collisions, so the limitation in effectiveness seems to ignore this fact.

The CAN bus calculations also made the assumption of a system with no collisions and that the system could use the available bandwidth of the CAN. In Konrad’s calculations, he showed that when using a 10MBps Ethernet the maximum number of FPS was lower than for a 1MBps CAN bus, he also showed when using a 100MBps system the maximum FPS value was three times greater than that of a 1MBps CAN bus [16].

3 Method

Section Configuration of test environment describes the configuration of the testing environment that will be used during the tests described in this thesis. Section Testing environments gives an overview of why each test was done and what describes the goal of each test. Section Tools Used For Network Testing introduces the reader to the tools used to produce the results and how performance was measured. Section Conducted tests describes each of the tests in detail.

3.1 Configuration of test environment

The laboratory environment is built around one custom built computer (Comp 1), one ASUS Laptop (Comp 2), and 2 Raspberry Pi single board computers (Comp 3 & Comp 4). The details of each of these computers are given in Table 3-1.

Table 3-1: Computers used in the different tests.

Computer	Operating System	Processor	Central Processing Unit Clock (MHz)	RAM (GB)
Comp 1	Win7 Professional Ver. 6.1.7601	AMD Phenom 2 x6 1000T	3300	8
Comp 2	Win7 Professional Ver. 6.1.7601	Intel Core i7 2670QM	2800	8
Comp 3	Raspbian Debian Wheezy Ver. 3.12	ARM1176JZF-S	700	0.5
Comp 4	Raspbian Debian Wheezy Ver. 3.12	ARM1176JZF-S	700	0.5

These computers are all connected to a ASUS RT-AC66U router running firmware 3.0.0.4.374_5517 as shown in Figure 3-1. These computers are connected to the router using CAT5e cables, the computers were configurd for 100Mbps with Full Duplex. The router is configured to support jumbo frames up to 9kB MTU, which is the largest acceptable value (for this router and most devices that accept jumbo frames). The router supports both generic Internet datagrams. The router also had the 2.4 GHz and 5 GHz Wi-Fi bands disabled. The router is then connected to the ISP network.

The highest link speed to the ISP net was measured to be 94 Mbps when connected to the Stockholm node of Bredbandskollen.se[1]. The ASUS router is connected with a CAT5e cable to the ISP.

The network interface cards of each computer is running with default settings except for the MTU value which has been increased on Comp 1 & Comp 2 to 9kB. Comp 3 & Comp 4 were configured for a 1.5kB MTU (which is the standard value).

[1] Bredbandskollen.se is a site operated by .SE (Stiftelsen för Internetinfrastruktur) for users to test their broadband network connections.

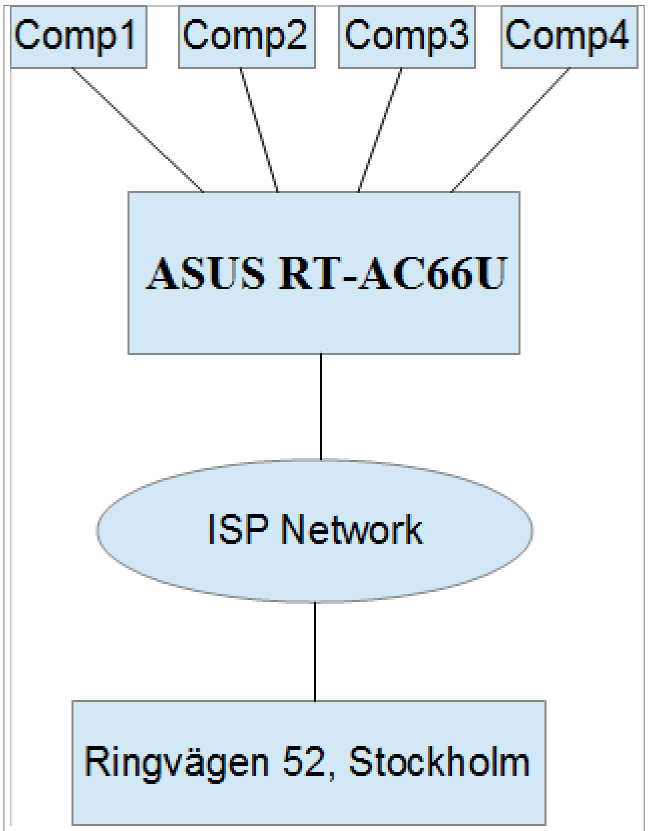


Figure 3-1: Overview of Test System

3.2 Testing environments

Traffic was sent in both the local area network and the ISP to hosts connected to the Internet.

3.2.1 Local Area Network(LAN)

The LAN described above will serve as a reference model to examine several different model networks or situations that might exist in an automobile communication system. One model could be where end-points need to be connected directly to each other because of time critical applications, this problem can be solved by using packet prioritization within Ethernet switches that supports this functionality. This is also known as Quality of Service (QoS). QoS is important for time critical applications to ensure that the data will reach its destination within a specific amount of time.

Class of Service (CoS) is a technique that is implemented by Ethernet to solve the time critical application problem. The relevant standard is IEEE 802.1D. Providing different priority to different frames is done by specifying in each Ethernet frame the priority of this frame. Note that this technique does not provide a *guarantee*. Additionally, this technique does not require any state to be remembered by a switch or router in the network. This technique is used to realize QoS as each frame that arrives at the switch or router is placed into an outgoing queue depending on the frame’s priority. However, many Internet Service Providers (ISPs) do not allow their users to specify any priority other than the default.

The data from the testing in the LAN model will provide a base for the tests in the Wide Area Network (WAN).

3.2.2 Wide Area Network(WAN)

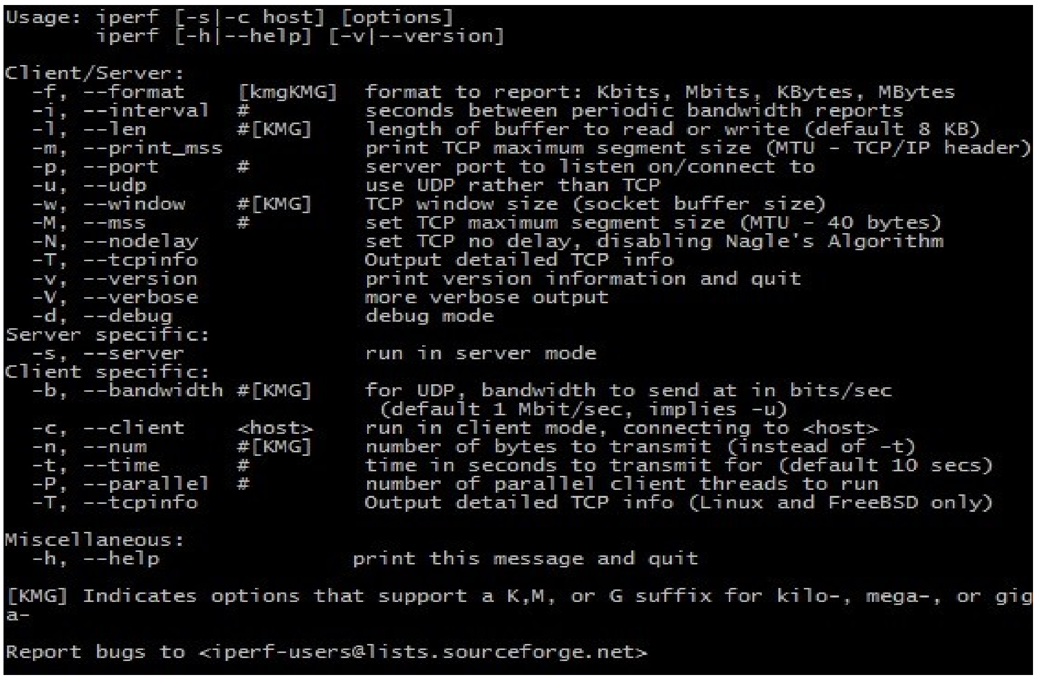
Tests with the WAN had the goal of evaluating the scalability of the network and examining how much a moderately congested network with multiple possible routes affects the data transfer rate. As mentioned in Section 3.3.1 it is not possible to set arbitrary CoS values to increase the value of a Ethernet frame as the ISP restricts this capability for customers. Before each of the tests there will be some network probes. The network probed using the ICMP protocol to determine how many hops there are between the two end-points and to measure the average delay. These probes were used to see if there was any congestion. Tests are spread out over 24 hours to determine how variable the delay between the end-points are.

3.3 Tools Used For Network Testing

Iperf is used to generate traffic and traffic is captured and analyzed using Wireshark. Each of these tools is describe in this section.

3.3.1 Iperf

Iperf is a cross-platform network traffic generating tool to measure throughput between end-points in a IP network using different transportation protocols. This tool can be used to test how well a given protocol's implementation works. The original Iperf version was written in the C programming language by the Distributed Applications Support Team (DAST). The software utilizes one machine to act as a server, then a client end-point runs Iperf to measure what the performance is. The program can be run with different settings to tune and optimize the testing (see the command line arguments to the program shown in Figure 3-2 – not that this is for the iperf3 3.0-BETA4 from 2 August 2010 and lack SCTP). The program can be run multiple times to get more accurate results.



```
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]

Client/Server:
  -f, --format    [kmgKMG]   format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #          seconds between periodic bandwidth reports
  -l, --len       #[KMG]     length of buffer to read or write (default 8 KB)
  -m, --print_mss            print TCP maximum segment size (MTU - TCP/IP header)
  -p, --port      #          server port to listen on/connect to
  -u, --udp                  use UDP rather than TCP
  -w, --window    #[KMG]     TCP window size (socket buffer size)
  -M, --mss       #          set TCP maximum segment size (MTU - 40 bytes)
  -N, --nodelay              set TCP no delay, disabling Nagle's Algorithm
  -T, --tcpinfo              Output detailed TCP info
  -v, --version              print version information and quit
  -V, --verbose              more verbose output
  -d, --debug                debug mode
Server specific:
  -s, --server               run in server mode
Client specific:
  -b, --bandwidth #[KMG]     for UDP, bandwidth to send at in bits/sec
                              (default 1 Mbit/sec, implies -u)
  -c, --client    <host>     run in client mode, connecting to <host>
  -n, --num       #[KMG]     number of bytes to transmit (instead of -t)
  -t, --time      #          time in seconds to transmit for (default 10 secs)
  -P, --parallel  #          number of parallel client threads to run
  -T, --tcpinfo              Output detailed TCP info (Linux and FreeBSD only)

Miscellaneous:
  -h, --help               print this message and quit

[KMG] Indicates options that support a K,M, or G suffix for kilo-, mega-, or gig
a-
Report bugs to <iperf-users@lists.sourceforge.net>
```

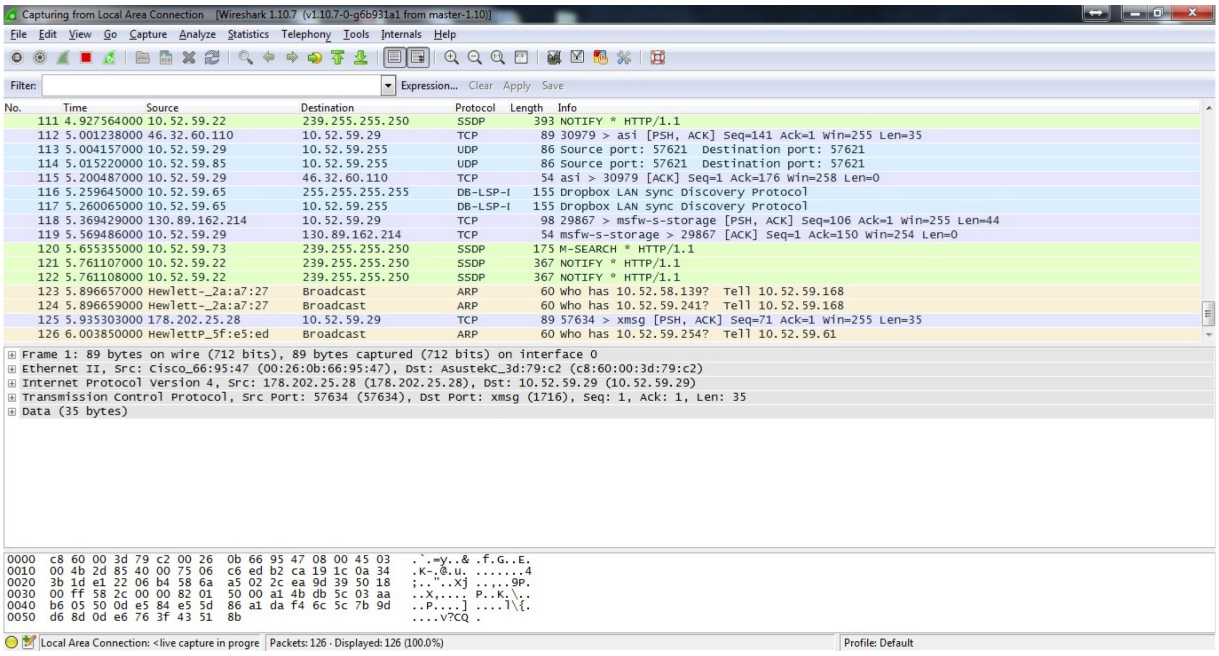
Figure 3-2: Iperf3 3.0-BETA4 command list

The original Iperf project stopped after version 2.0.5 in July 2010. Since then Iperf has been rewritten from scratch leading to a version called Iperf3. Iperf3 works on the same principle as Iperf and many of the functions have been developed to also work in Iperf3. Iperf3 also has support for SCTP. Unfortunately, Iperf3 does not support older versions of Iperf [18, p. 3].

Iperf uses TCP and UDP in different manners. TCP tests can measure the network bandwidth between two end-points. UDP test can be used to determine network jitter.

3.3.2 Wireshark

Wireshark was developed by Gerald Combs in the late 1990s, it works as a GUI to TCPdump. This makes traffic analysiseasier and more understandable for both network professionals and for people whom are not too familiar with command line programs. It also uses the libpcap library to capture the frames received by a network interface (or multiple network interfaces). One of the strengths when analyzing frames with Wireshark is the ease of applying filters as the environment is graphical. An example of Wireshark's GUI is shown in Figure 3-3.


Capturing from Local Area Connection [Wireshark 1.10.7 (v1.10.7-0-g6b931a1 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.      Time        Source            Destination       Protocol  Length  Info
    111 4.927564000 10.52.59.22       239.255.255.250   SSDP       393 NOTIFY * HTTP/1.1
    112 5.001238000 46.32.60.110      10.52.59.29       TCP         89 30979 > asi [PSH, ACK] Seq=141 Ack=1 Win=255 Len=35
    113 5.004157000 10.52.59.29       10.52.59.255      UDP         86 Source port: 57621  Destination port: 57621
    114 5.015220000 10.52.59.85       10.52.59.255      UDP         86 Source port: 57621  Destination port: 57621
    115 5.200487000 10.52.59.29       46.32.60.110      TCP         54 asi > 30979 [ACK] Seq=1 Ack=176 Win=258 Len=0
    116 5.259645000 10.52.59.65       255.255.255.255   DB-LSP-I   155 Dropbox LAN sync Discovery Protocol
    117 5.260065000 10.52.59.65       10.52.59.255      DB-LSP-I   155 Dropbox LAN sync Discovery Protocol
    118 5.369429000 130.89.162.214    10.52.59.29       TCP         98 29867 > msfw-s-storage [PSH, ACK] Seq=106 Ack=1 Win=255 Len=44
    119 5.569486000 10.52.59.29       130.89.162.214    TCP         54 msfw-s-storage > 29867 [ACK] Seq=1 Ack=150 Win=254 Len=0
    120 5.655355000 10.52.59.73       239.255.255.250   SSDP       175 M-SEARCH * HTTP/1.1
    121 5.761107000 10.52.59.22       239.255.255.250   SSDP       367 NOTIFY * HTTP/1.1
    122 5.761108000 10.52.59.22       239.255.255.250   SSDP       367 NOTIFY * HTTP/1.1
    123 5.896657000 Hewlett-_2a:a7:27 Broadcast         ARP         60 Who has 10.52.58.139?  Tell 10.52.59.168
    124 5.896659000 Hewlett-_2a:a7:27 Broadcast         ARP         60 Who has 10.52.59.241?  Tell 10.52.59.168
    125 5.935303000 178.202.25.28     10.52.59.29       TCP         89 57634 > xmsg [PSH, ACK] Seq=71 Ack=1 Win=255 Len=35
    126 6.003850000 HewlettP_5f:e5:ed Broadcast         ARP         60 Who has 10.52.59.254?  Tell 10.52.59.61

⊞ Frame 1: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
⊞ Ethernet II, Src: Cisco_66:95:47 (00:26:0b:66:95:47), Dst: AsustekC_3d:79:c2 (c8:60:00:3d:79:c2)
⊞ Internet Protocol Version 4, Src: 178.202.25.28 (178.202.25.28), Dst: 10.52.59.29 (10.52.59.29)
⊞ Transmission Control Protocol, Src Port: 57634 (57634), Dst Port: xmsg (1716), Seq: 1, Ack: 1, Len: 35
⊞ Data (35 bytes)

0000  c8 60 00 3d 79 c2 00 26  0b 66 95 47 08 00 45 03   .`.=y..& .f.G..E.
0010  00 4b 2d 85 40 00 75 06  c6 ed b2 ca 19 1c 0a 34   .K-.@.u. .......4
0020  3b 1d e1 22 06 b4 58 6a  a5 02 2c ea 9d 39 50 18   ;.."..Xj ..,..9P.
0030  00 ff 58 2c 00 00 82 01  50 00 a1 4b db 5c 03 aa   ..X,.... P..K.\..
0040  b6 05 50 0d e5 84 e5 5d  86 a1 da f4 6c 5c 7b 9d   ..P....] ....l\{.
0050  d6 8d 0d e6 76 3f 43 51  8b                        ....v?CQ .

Local Area Connection: <live capture in progre | Packets: 126 · Displayed: 126 (100.0%) | Profile: Default
```

Figure 3-3: Wireshark GUI with an open session

Wireshark also supports *promiscuous mode*. In this mode the network interface card (NIC) or Wireless NIC (WNIC) capture all of the frames that the interface receives, rather than those only intended for the intended interface. This does not guarantee that every frame is captured. If all frames must be captured, then port mirroring and network taps are examples of technologies that can be used to do this.

The filter "icmp && ip.src == 74.125.232.119" will filter out from those packets that Wireshark has collected only ICMP packets from a specific IP address, in this example the response packets from a ping request to the site http://www.google.se.

Wireshark can be used to visualize the throughput behavior of a connecting/association with the IO tool. An example of this is shown in Figure 3-4. This graph was generated by this IO tool and it shows ICMP request and replay, the each stages in the graph shows the increases in rate of ICMP packets. Each time the graph increases and flattens represents a new ping session running concurrently with the previous active ping sessions. Towards the end of the graph the rate reaches zero indicates that all ping sessions have terminated.

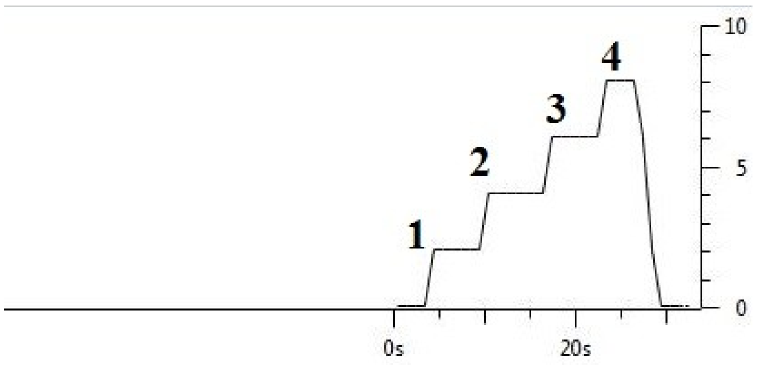


Figure 3-4: ICMP throughput graph

Table 3-2 shows the different sizes of payloads used during each new ping session.

Table 3-2: Payload sizes for each stage

Stage	Ping(ICMP) Payload (bytes)
Stage 1	1400
Stage 2	1300
Stage 3	1200
Stage 4	1100

3.4 Conducted tests

This section describes the tests that were conducted.

3.4.1 Jumbo frame tests

Tests with different size frames were made in order to see how throughput in both the number of frames and the amount of data can be improved with the use of jumbo frames. In order to make sure that the tests are as accurate as possible they were conducted between Comp1 and Comp2. These two computers were used as they had more memory and computational capacity than the Raspberry Pis. The tests used different versions of iperf in order to avoid any software limitations or non-optimal implementations of the tests. These tests will determine whether the transfer rate is notably higher than when using the standard MTU value of 1.5kB.

3.4.2 Smallest Possible Frame Tests

Smallest possible frame tests monitor latency differences when packets have a very small payload. The goal is to see if small payloads ensure a more stable performance, for example for time-critical applications where low transmission delay and rapid processing are required in order to ensure that the data is still valid when it reaches its destination.

3.4.3 Bi-directional Congestion Tests

Bi-directional congestion tests examine how throughput changes when all devices in the network are communicating to each other without limits. In these tests all devices in the LAN are running their own respective servers and each a client connecting to all other devices in the network. All Ethernet frames are 1500 bytes in size, the standard MTU value. These tests examine whether several devices can communicate without having to wait for access to the bus (or network) in order to send their data.

3.4.4 Latency Difference Tests

Latency difference tests were run between Comp1 and Comp2 using the OS utility ping. The goal of this test to see how quickly data can be transferred between end-points in a small LAN environment and to determine if Ethernet can serve as a possible replacement for a CAN for reliable time-critical applications.

3.4.5 IPv4 versus IPv6 Performance Difference Tests

IPv6 is the recommended version of IP, because it is the IP standard of the future. Unfortunately, as of today not all ISPs provide IPv6 addresses to their customers. The ASUS RT-AC66U router used in these tests solves this by providing different tunneling methods such as 6to4 (one of the more popular tunneling techniques). This method encapsulates the IPv6 datagram inside an IPv4 datagram in order to transport it over an IPv4 network until it reaches a point where native IPv6 support is available.

The goal of the IPv4 versus IPv6 Performance Difference tests are to examine the performance of IPv6 implemented in customer grade hardware and to see if this provides a performance increase for IPv6 in comparison with IPv4 with respect to throughput and delay. These tests will be done both in local and wide area network.

3.4.6 Monitoring Latency In A Network

Tests will be performed to nonitoring latency within a network by using IPv4 ICMP Ping requests from Comp3 to another Raspberry Pi at Ringvägen 52, Stockholm where the Swedish company MTG has their office. This company is using these Raspberry Pis during their normal office hours to view statistics, but these systems were unused during the weekends when these tests were conducted. The tests measure how latency of the network path between these Raspberry Pis changed during a period of 24 hours. Comp3 continuously pinged the Raspberry Pi located at Ringvägen once each second for 24 hours.

4 Results

In this chapter, all of the data collected during the tests are analyzed. The section headings indicate what test results are being analyzed. Section 4.1 considers the results of tests in the LAN environment, while Section 4.2 considers the tests in the WLAN environment.

The organization of each section is as follows. The section heading indicates the relevant type of test. In each section tables and graphs show the measured throughput. The X-axis represents time and the Y-axis represents the number of frames sent at that specific time. Each pixel in the graph represents a second. The tables are divided into subsections of 5 seconds give an average throughput value. This average throughput provides the reader with an estimate of the performance in the specific test's setting. The frame size indicates the MTU, thus a 9kB frame size corresponds to an MTU of 9kB.

4.1 LAN Results

Section 4.1 starts with the results of tests in the LAN environment. It first shows the effect of different values of MTU (*Jumbo frames*). Then it shows the results gathered when several different devices are using the router at the same time.

4.1.1 Jumbo Frame Results

In this section the results of throughput are shown as a function of MTU in to Table 4-9. The variaiton in throughput with time is shown in the graphs in Figure 4-1 to Figure 4-9.

Table 4-1: Comp 1→2 IPv4 - 9K MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	199	334
05.00-10.13	234	392
10.13-15.24	204	342
15.24-20.25	231	388
20.25-25.41	214	219
00.00-30.10	~1310	376

Table 4-2: Comp 1→2 IPv4 - 8K MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	290	487
05.00-10.00	276	463
10.00-15.05	246	412
15.05-20.06	226	378
20.06-25.06	260	436
00.00-30.00	~1550	443

Table 4-3: Comp 1→2 IPv4 - 7kB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.03	184	307
05.03-10.04	254	427
10.04-15.04	205	344
15.04-20.04	222	372
20.04-25.14	243	400
00.00-30.00	~1220	350

Table 4-4: Comp 1→2 IPv4 - 6kB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	166	278
05.00-10.00	165	277
10.00-15.00	108	181
15.00-20.02	145	243
20.02-25.02	130	217
00.00-30.00	885	247

Table 4-5: Comp 1→2 IPv4 - 5kB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.18	144	233
05.18-10.32	146	237
10.32-15.32	149	250
15.32-20.33	148	247
20.33-25.36	129	215
00.00-30.18	816	227

Table 4-6: Comp 1→2 IPv4 - 4kB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	190	318
05.00-10.00	186	312
10.00-15.07	236	391
15.07-20.07	230	385
20.07-25.07	241	404
00.00-30.00	~1310	375

Table 4-7: Comp →2 IPv4 - 3kB MTU

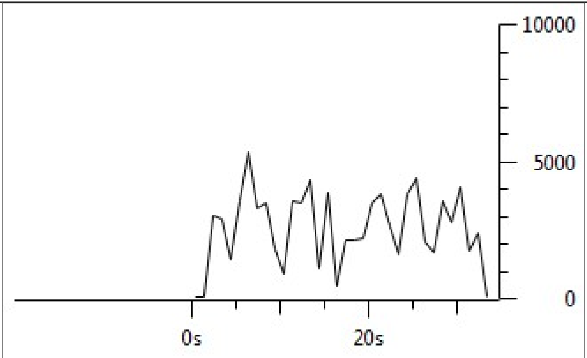
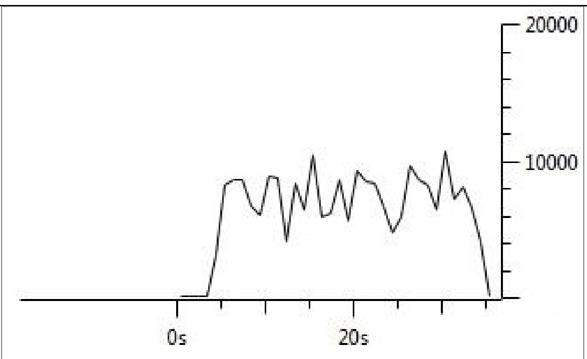
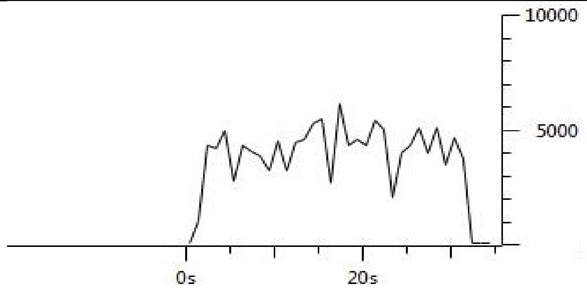
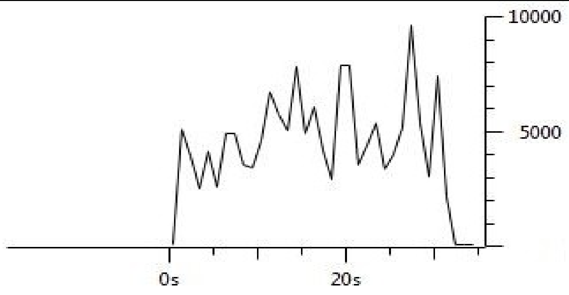
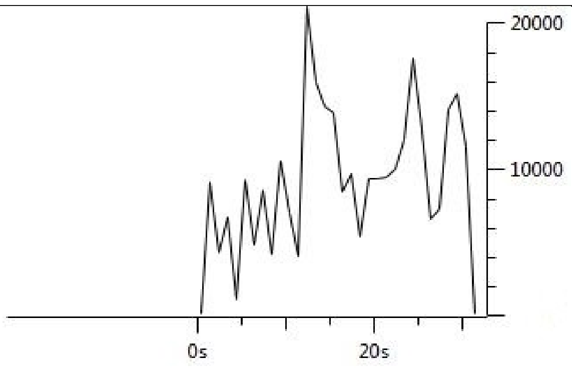
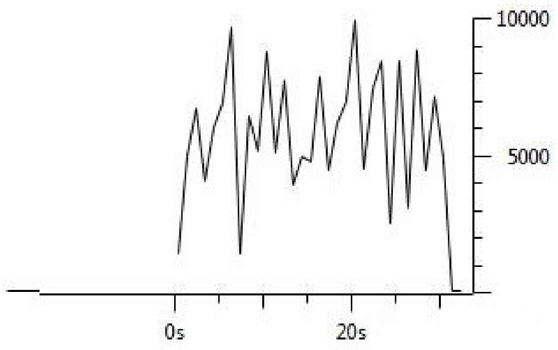
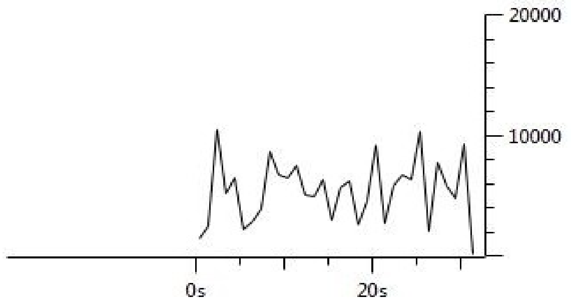
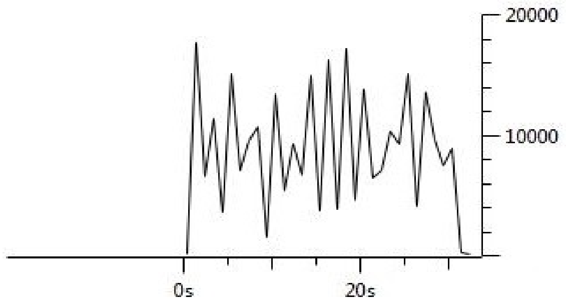
Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	256	430
05.00-10.00	245	411
10.00-15.00	297	498
15.00-20.00	394	662
20.00-25.00	283	474
00.00-30.00	~1710	488

Table 4-8: Comp 1→2 IPv4 - 2KB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.00	302	506
05.00-10.15	329	535
10.15-15.27	260	426
15.27-20.27	301	504
20.27-25.27	317	532
00.00-30.03	~1770	507

Table 4-9: Comp 1→2 IPv4 - 1.5kB MTU

Time (sec)	Transferred Data (MByte)	Throughput (Mbps)
00.00-05.18	210	341
05.18-10.18	191	320
10.18-15.18	205	344
15.18-20.18	346	581
20.18-25.18	284	477
00.00-30.00	~1480	424

 **Figure 4-1: IPv4 Comp1->Comp2 9kB MTU**	 **Figure 4-2: IPv4 Comp1->Comp2 8kB MTU**
 **Figure 4-3: IPv4 Comp1→Comp2 7kB MTU**	 **Figure 4-4: IPv4 Comp1→Comp2 6kB MTU**
 **Figure 4-5: IPv4 Comp1→Comp2 5kB MTU**	 **Figure 4-6: IPv4 Comp1→Comp2 4kB MTU**
 **Figure 4-7: IPv4 Comp1→Comp2 3kB MTU**	 **Figure 4-8: IPv4 Comp1→Comp2 2kB MTU**

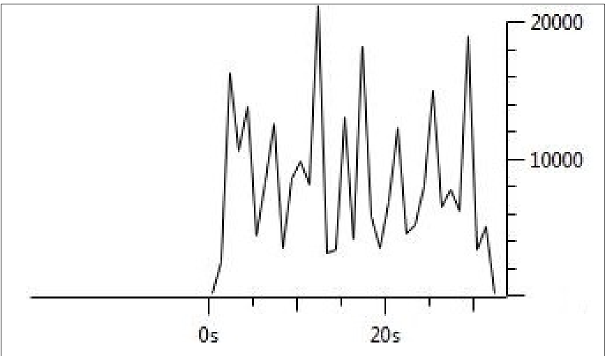
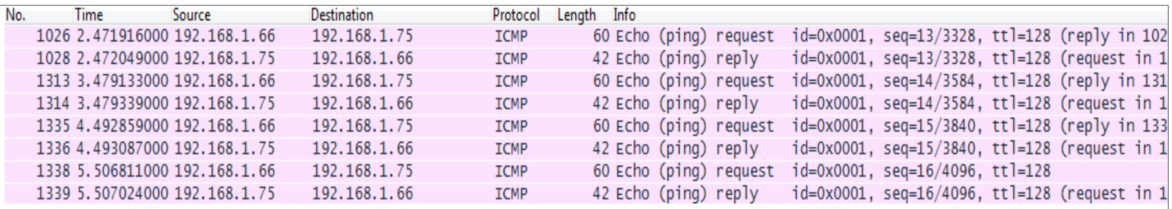


Figure 4-9: IPv4 Comp1→Comp2 1.5kB MTU

4.1.2 Smallest Frame & Latency Results

Figure 4-10 shows the output of Wireshark when sending IPv4 ICMP ping requests with a payload size of 0 bytes. The resulting times are shown in Table 4-10. This is followed by the corresponding results for IPv6 in Figure 4-11 and Table 4-11. This is followed by similar IPv6 results when the destination is the router, in Figure 4-12 and Table 4-12.

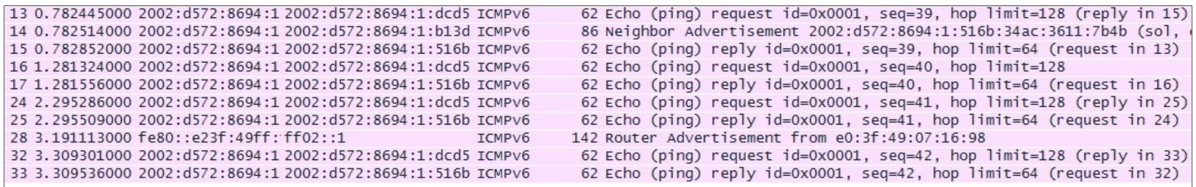


```
No.     Time        Source        Destination    Protocol Length Info
   1026 2.471916000 192.168.1.66  192.168.1.75   ICMP       60 Echo (ping) request  id=0x0001, seq=13/3328, ttl=128 (reply in 102
   1028 2.472049000 192.168.1.75  192.168.1.66   ICMP       42 Echo (ping) reply    id=0x0001, seq=13/3328, ttl=128 (request in 1
   1313 3.479133000 192.168.1.66  192.168.1.75   ICMP       60 Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 131
   1314 3.479339000 192.168.1.75  192.168.1.66   ICMP       42 Echo (ping) reply    id=0x0001, seq=14/3584, ttl=128 (request in 1
   1335 4.492859000 192.168.1.66  192.168.1.75   ICMP       60 Echo (ping) request  id=0x0001, seq=15/3840, ttl=128 (reply in 133
   1336 4.493087000 192.168.1.75  192.168.1.66   ICMP       42 Echo (ping) reply    id=0x0001, seq=15/3840, ttl=128 (request in 1
   1338 5.506811000 192.168.1.66  192.168.1.75   ICMP       60 Echo (ping) request  id=0x0001, seq=16/4096, ttl=128
   1339 5.507024000 192.168.1.75  192.168.1.66   ICMP       42 Echo (ping) reply    id=0x0001, seq=16/4096, ttl=128 (request in 1
```

Figure 4-10: ICMP Ping Session Comp1→Comp2 with 0 byte payload

Table 4-10: ICMP 0 byte payload latency

(Seconds)	Request 1	Request 2	Request 3	Request 4
Sent	2.471916	3.479133	4.492859	5.506811
Response	2.472049	3.47933	4.493087	5.507024
Difference	0.000133	0.000197	0.000228	0.000213

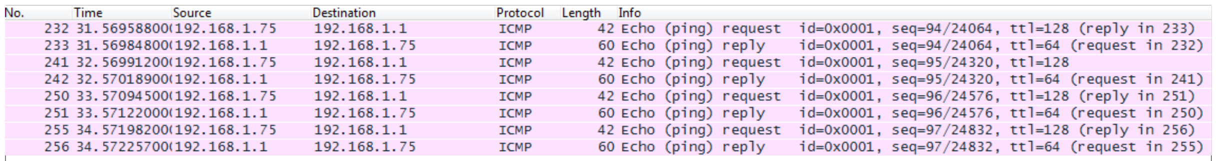


```
13 0.782445000 2002:d572:8694:1 2002:d572:8694:1:dcd5 ICMPv6      62 Echo (ping) request id=0x0001, seq=39, hop limit=128 (reply in 15)
14 0.782514000 2002:d572:8694:1 2002:d572:8694:1:b13d ICMPv6      86 Neighbor Advertisement 2002:d572:8694:1:516b:34ac:3611:7b4b (sol,
15 0.782852000 2002:d572:8694:1 2002:d572:8694:1:516b ICMPv6      62 Echo (ping) reply id=0x0001, seq=39, hop limit=64 (request in 13)
16 1.281324000 2002:d572:8694:1 2002:d572:8694:1:dcd5 ICMPv6      62 Echo (ping) request id=0x0001, seq=40, hop limit=128
17 1.281556000 2002:d572:8694:1 2002:d572:8694:1:516b ICMPv6      62 Echo (ping) reply id=0x0001, seq=40, hop limit=64 (request in 16)
24 2.295286000 2002:d572:8694:1 2002:d572:8694:1:dcd5 ICMPv6      62 Echo (ping) request id=0x0001, seq=41, hop limit=128 (reply in 25)
25 2.295509000 2002:d572:8694:1 2002:d572:8694:1:516b ICMPv6      62 Echo (ping) reply id=0x0001, seq=41, hop limit=64 (request in 24)
28 3.191113000 fe80::e23f:49ff: ff02::1                ICMPv6     142 Router Advertisement from e0:3f:49:07:16:98
32 3.309301000 2002:d572:8694:1 2002:d572:8694:1:dcd5 ICMPv6      62 Echo (ping) request id=0x0001, seq=42, hop limit=128 (reply in 33)
33 3.309536000 2002:d572:8694:1 2002:d572:8694:1:516b ICMPv6      62 Echo (ping) reply id=0x0001, seq=42, hop limit=64 (request in 32)
```

Figure 4-11: ICMPv6 Ping Session snippet Comp1-Comp2 with 0 byte payload

Table 4-11: ICMPv6 with 0 byte payload latency

(Seconds)	Request 1	Request 2	Request 3	Request 4
Sent	0.782445	1.281324	2.295286	3.309301
Response	0.782852	1.281556	2.295509	3.309536
Difference	0.000407	0.000232	0.000223	0.000235



```
No.    Time         Source         Destination    Protocol  Length Info
   232 31.56958800(192.168.1.75   192.168.1.1    ICMP          42 Echo (ping) request  id=0x0001, seq=94/24064, ttl=128 (reply in 233)
   233 31.56984800(192.168.1.1    192.168.1.75   ICMP          60 Echo (ping) reply    id=0x0001, seq=94/24064, ttl=64 (request in 232)
   241 32.56991200(192.168.1.75   192.168.1.1    ICMP          42 Echo (ping) request  id=0x0001, seq=95/24320, ttl=128
   242 32.57018900(192.168.1.1    192.168.1.75   ICMP          60 Echo (ping) reply    id=0x0001, seq=95/24320, ttl=64 (request in 241)
   250 33.57094500(192.168.1.75   192.168.1.1    ICMP          42 Echo (ping) request  id=0x0001, seq=96/24576, ttl=128 (reply in 251)
   251 33.57122000(192.168.1.1    192.168.1.75   ICMP          60 Echo (ping) reply    id=0x0001, seq=96/24576, ttl=64 (request in 250)
   255 34.57198200(192.168.1.75   192.168.1.1    ICMP          42 Echo (ping) request  id=0x0001, seq=97/24832, ttl=128 (reply in 256)
   256 34.57225700(192.168.1.1    192.168.1.75   ICMP          60 Echo (ping) reply    id=0x0001, seq=97/24832, ttl=64 (request in 255)
```

Figure 4-12: IPv4 ICMP Ping Session to ASUS RT-AC66U router

Table 4-12: ICMP 0 byte payload to Router

(Seconds)	Request 1	Request 2	Request 3	Request 4
Sent	31.569588	32.569912	33.547945	34.571982
Response	31.569848	32.570189	33.57122	34.572257
Difference	0.00026	0.000277	0.023275	0.000275

4.1.3 Bi-directional Congestion Results

Table 4-13: Comp1→Comp2 Congested Network 1.5kB MTU

Time (sec)	Transferred Data(MByte)	Throughput (Mbps)
00.0- 05.0	131	220
05.0-10.0	132	221
10.0-15.0	134	224
15.0-20.0	131	219
20.0-25.0	134	224
00.00-30.00	794	222

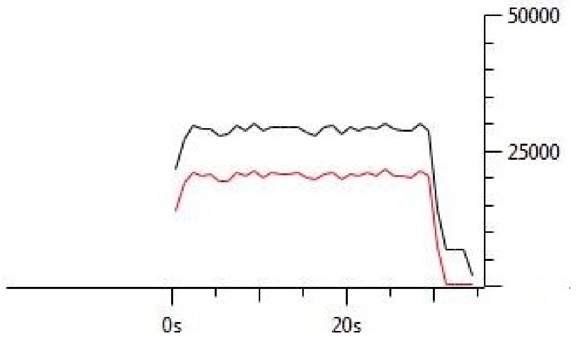


Figure 4-13: Number of Received and transmitted frames

red = transmitted frames from comp1

black = all received frames from comp2-4

4.1.4 IPv4 versus IPv6 Performance Results

Table 4-14 and Table 4-15 shows the throughput results obtained with jperf2.0.2 with a 1.5kB MTU.

Table 4-14: Comp1→Comp2 IPv6 1.5kB MTU

Time (sec)	Transferred Data(MByte)	Throughput (Mbps)
0.0- 5.0	158	266
5.0-10.0	155	259
10.0-15.0	154	258
15.0-20.0	155	261
20.0-25.0	159	266
0.00-30.00	936	262

Table 4-15: Comp1→Comp2 IPv4 1.5kB MTU

Time (sec)	Transferred Data(MByte)	Throughput (Mbps)
0.0- 5.0	134	224
5.0-10.0	133	223
10.0-15.0	135	227
15.0-20.0	134	224
20.0-25.0	134	224
0.00-30.00	798	223

4.2 WAN Results

The ping statistics for the Raspberry Pi which was pinged every second for a period of 24 hours are:

```
   94.246.93.230 ping statistics
86400 packets transmitted, 86372 received, 0% packet loss, time 86518907ms
rtt min/avg/max/mdev = 2.413/2.888/32.583/0.633 ms
```

A traceroute of the path between from Comp3 to the computer pinged in the results shown above was:

```
    1   <10 ms   <10 ms   <10 ms  router.asus.com [192.168.1.1]

     2   <10 ms     1 ms   <10 ms  ua-213-114-128-
1.cust.bredbandsbolaget.se [213.114.128.1]

     3     1 ms     2 ms     1 ms  ti3001d400-xe6-0-1.ti.telenor.net
[146.172.107.237]

     4     1 ms     1 ms     1 ms  ti3001c400-ae1-0.ti.telenor.net
[146.172.99.69]

     5     1 ms     1 ms     1 ms  ti3002b400-ae1-0.ti.telenor.net
[146.172.105.82]

     6     1 ms     1 ms     1 ms  netnod-ix-ge-a-sth.ip-only.net
[194.68.123.92]

     7     1 ms     1 ms     1 ms  62.109.44.161

     8    22 ms     1 ms     1 ms  62.109.44.170

     9     2 ms     2 ms     2 ms  sesto0001-rc3.ip-only.net
[213.132.112.82]

    10     6 ms     3 ms     6 ms  94.246.88.98

    11     2 ms     2 ms     2 ms  94.246.93.230
```

Another Raspberry Pi was pinged every 5 minutes for a period of 24 hours, leading to the following statistcs:

```
   88.131.87.100 ping statistics
290 packets transmitted, 281 received, +7 errors, 3% packet loss, time 24
hours
RTT min/avg/max/mdev = 2.360/2.606/3.725/0.203 ms
```

A traceroute of the route between from Comp3 to the computer pinged above was:

```
   1   <10 ms   <10 ms   <10 ms  router.asus.com [192.168.1.1]

     2     1 ms   <10 ms   <10 ms  ua-213-114-128-
1.cust.bredbandsbolaget.se [213.114.128.1]

     3     1 ms     1 ms     1 ms  ti3001d400-xe6-0-1.ti.telenor.net
[146.172.107.237]

     4     1 ms     1 ms     1 ms  ti3001c400-ae1-0.ti.telenor.net
[146.172.99.69]

     5     1 ms     1 ms     1 ms  ti3001b400-ae0-0.ti.telenor.net
[146.172.105.25]

     6     5 ms     1 ms     2 ms  tdc-1.ti.telenor.net [148.122.8.214]

     7     1 ms     1 ms     1 ms  te2-1-2549.kst-pe2.sto.se.ip.tdc.net
[213.50.210.41]

     8     2 ms     2 ms     2 ms  213.50.210.42
```

```
  9     2 ms     2 ms     2 ms  88.131.87.100
```

5 Analysis

In this section I will analyze the data that was captured during my tests, the results will be explained and any factors that could have effected these results will be mentioned.

5.1 Jumbo Frame Result Analysis

When testing different MTU values we can observe that when using Microsoft's Windows 7 and the specific firmware of the ASUS router that the most optimal MTU value was 2 kB which is only 0.5 kB larger than the standard value. This may be due to limitations in either hardware or the software, but we can see that the throughput when the MTU reaches 8kB is again ~500Mbps. The lower performance between 2 kB and 8 kB may indicate that there is a poor implementation of jumbo frames by one or more of these devices. It is important to note that when using different values for MTU fewer frames were sent over the network when using a larger MTU (as the same total amount of data as transferred for each different MTU value).

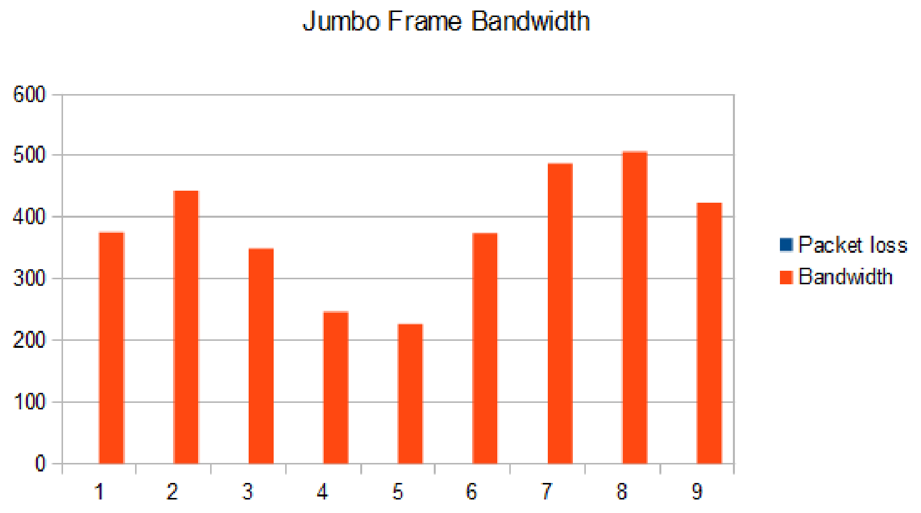


Figure 5-1: Throughput as a function of MTU

I did not expect these results when I set out to examine the performance of jumbo frames. I expected that growth in performance would increase as the MTU increased. Some factors that I think could be the reason behind not achieving this result is that the jumbo frame implementation was not optimal in the hardware or software that I used in my tests. Additionally, the cable quality could have been a reason for less than optimal performance. For example, if CAT6 cables were used the performance might have been increased, I only checked for missing TCP segments when checking for errors during transmissions. There could also have been some other programs running that transmitted frames, thus not all of the received frames were sent by the sending node in these tests.

Because using a larger MTU allows fewer frames to send a given amount of traffic, if the hardware can process a 9kB frame in the same amount of time as a 1.5kB frame, then fewer frames should lead to less congestion. However, this ignores the fact that it takes a fixed amount of time to send each bit of each frame, thus both sending and receiving larger frames does take longer – irrespective of how long it takes the switch, router, or other device to process the frame.

During my tests my throughput values were much lower than what Arjun Reddy Kanthla reported in his thesis project, I believe that my results were restricted due to cabling quality and because of the use of different OS. When reading his thesis I could also see he used an NIC which reduced computation by the CPU to achieve a better performance. Considering I did not used an NIC card, hence all the computation was done by my CPU which was also being used by OS and other services running in the background. This could have effected the results considering my measured values never reached the 900Mbps which he reported.

5.2 Smallest Frame Result Analysis

From the test results shown in Table 5-1, we can see that IPv6 has an median increase in delay between Comp1 and Comp2 of 0.029 milliseconds. However, in this table we see that IPv6 for request 3 was faster than IPv4, but the general performance difference is that there is a slight increase in latency when it comes to sending packets with a 0 byte payload. This result had the same direction in terms of performance gain as expected. The IPv6 header is 40 bytes long while the IPv4 minimum header is 20 bytes long with an arbitrary n bytes long option section, but the minimum header length is 20 bytes. However this was only 4 ICMP requests done and the results can't show how the 2 protocol's perform compared to each other. But we can see that the difference between the protocol's are very small.

Table 5-1: Latency Difference of IPv6 and IPv4

	Request 1 (seconds)	Request 2 (seconds)	Request 3 (seconds)	Request 4 (seconds)
IPv4	0.000133	0.000197	0.000228	0.000213
IPv6	0.000407	0.000232	0.000223	0.000235
Difference	+0.000274	+0.00035	-0.0005	+0.00022

The IPv6 header is a fixed size while the IPv4 header is not a fixed size as state above. IPv6 specified a fixed size header in order for hardware to work more efficiently, since the header will always be 40 bytes. However, the hardware used in these tests did not have support for only IPv6 so there was an expected difference between the protocols – the expected difference was a 20 byte difference and with a link data rate of 1000Mbps – it should have taken $2*1.6x10^{-7}$ seconds (i.e., 0.00000016 s) longer to send the IPv6 frame. As the median difference was $-2.9x10^{-5}$ – we can see that the measured difference was nearly 100 times larger than what we expected.

A series of ICMP packets were send to the ASUS router to try to determine if there was a noticeable difference due to the performance of the router. Unsurprisingly the ping results had a higher delay than when we sent an ICMP request through this router. This was because that when the router is forwarding a packet to another host the router does not have to inspect the packet to determine what to do with it, it simply forwards the packet. Since the router was the destination of the ICMP packet, the router had to inspect the packet, understand what the ICMP request wanted the router to do, and then respond back to the source node.

Interestingly, we can see that IPv6 had a higher throughput when compared to IPv4 when using the standard 1.5kB MTU. This result was unexpected because most tests done by others indicated that IPv4 was at times slightly faster than IPv6 was. However, we should note that our measurements of the IPv4 was more stable that the IPv6 throughput. This is perhaps expected because IPv4 code is more mature that the IPv6 code (due to their difference in ages). The OS and other applications results could have affected the throughput results (as stated earlier) as we did not explicitly turned off all of the unnecessary services on each of the devices that were being used for these tests. However, if that were the case I would have expected the values to fluctuate a lot more. Despite this, the IPv6 throughput during the test was *always* higher than the IPv4 throughput.

However, these tests did not evaluate the use of IPsec authentication and encryption. These tests showed that the raw unencrypted IPv6 throughput was greater than that of IPv4, but that IPv6 had a slight higher latency.

5.3 Bi-directional Result Analysis

During these tests we could see that the throughput between Comp1 and Comp2 was unchanged from the jumbo frame tests. These two computers were both configured for 9kB MTU and the average throughput from this test are 10Mbps faster than the result from the jumbo frame test. This difference could be a result from other services as mentioned earlier that could have effected the performance of the test. However since that the ASUS router and the computers were both configured to be Full-Duplex this result was to be expected.

5.4 WAN Connectivity and Latency

We could see that the ping requests varied between 2 ms up to 32 ms, this shows that either the network was congested or the path that ICMP request took was much longer for the 32 ms instance than when using the 2 ms path. If these were the latency values within a vehicle then the data received could not be trusted, but this path is much farther than packets would ever go in any automobile. However, these value might be suggestive of how a realistic congested network might affects packet delays. However, in an automobile application congestion is **not** expected, hence the delays should not fluctuate much. We can also see via the trace route command that the paths were very different even though the Raspberry Pis were located close to each other physically. Further comments will be given about this in Section 6.1.

6 Conclusions and Future work

In this Chapter I first present those conclusions I have drawn from my result analyze and also comment on topics where I feel more research is needed to increase performance for Ethernet to serve as a potential candidate. I will also state what I would do differently if I had to redo this thesis project from the start. Section 6.2 will describe those fields that I feel are most important for further research and work that might be done in order to achieve better performance for the target communication system. Lastly in section 6.3 I will mention environmental effects that using a Ethernet based system could have and how use of such systems can provide secure communication for vehicles in traffic.

6.1 Conclusions

During my tests of Ethernet in terms of latency and throughput results I can draw the conclusion that Ethernet is a suitable candidate to replace CAN systems even when using customer grade equipment. I draw this conclusion based on the CPAC system requirements which Muhammad Ibrahim was able to meet with a industrial grade Ethernet switch [17]. In his conclusion he mentions that the delay requirement to be met for a Ethernet based network must be lower than 1ms. He also mentions that as more devices are added that require higher throughput that replacing CAN with Ethernet is increasingly necessary. In Joakim Sandberg's thesis he mentions a lot of different devices such as radars and cameras that can be used in order to achieve platooning [18]. In order to see a high quality picture with good resolution from a camera attached to the rear bumper higher bandwidth is needed than can be provided by CAN. Ethernet therefore can serve as a great replacement for CAN since the delay requirement is met and devices such as cameras and other devices that require higher bandwidth can be used *without* the limitations that would be caused by the low bandwidth that a CAN network would have.

I have also learned that the IPv6 adoption is not yet as successful as the IPv4 adoption. This is most likely because IPv6 is a newer standard than IPv4 and native IPv6 support is not yet available for all customers. However, in my ping tests I had less than 1ms responce times in my ping tests, therefore demonstrating that IPv6 can be successfully used over IPv4. I also believe that as more research is done that IPv6 performance will increase.

When I conducted my WAN tests, I pinged two hosts located at Ringvägen 52 in Stockholm from my home. From these tests, I found that some ping requests were lost (in the case of `88.131.87.100` there was 3% packet loss, while in the case of `94.246.93.230` *only* 0.03% of packets were lost). Additionally, I saw that the RTT value from some pings were up to `32.58` ms, but from the standard deviation it can be see that such long delays were quite rare. Many factors could cause this delay. One could be that the network was congested, hence my packets never made it to its destination and were dropped among the way or the packet was corrupted in during transfer. Despite the fact that these two Raspberry Pis where located close to each other and were connected to the same physical switch, they had different network addresses and the paths from my test computer to these two hosts were different. We might even ping the same host but via two different IP addresses, hence different routes could be used.

The results from the WAN tests showed me that if Ethernet is to be used as a replacement for CAN then each device must be given a specific IP address and that the whole system must be appropriately configured. The use of static IP addresses is not encourage as this makes configuring the network more tedious and potentially requires a lot of time. DHCP can provides an IP address dynamically, reducing the amount of effort for configuration when connecting a new node to the network. However, for security purposes in a vehicle if a node is given an arbitrary IP address, then the other components must be informed of this so that devices within the vehicle can continue to communicate.

I have also learned that jumbo frames have a great potential in order to reduce congestion by sending fewer frames. However, in my test environment I could not see any speedup with different MTU values, this could be due to inefficient processing of jumbo frames in the ASUS RT-AC66U router, but it could also be a global problem. When I pinged Google’s web server with a payload of 9000 with the option to not fragment the packet I received 100% loss rate – as the MTU of the link to my ISP had a MTU of 1500. This result gave me the insight that even if jumbo frames have great potential for a large network that they are not yet widely utilized. Therefore, I can draw the conclusion that jumbo frames have not received enough research or deployment. I also note that the Ethernet standard says that the highest recommended MTU value should be 1500 bytes, which is not a jumbo frame.

I have also learned that one of the downsides of CAN is that only one device can use the bus at one point in time, while one device is using the bus the other devices have to patiently wait for the bus to become free. With my bi-directional tests I could see that even if I tried to create congestion at Comp1 that I was still able to communicate with the other nodes at an unhindered speed. This insight suggests to me that even if there are many devices in a network that are all sending traffic, all devices could send their messages to each other only being limited by the bandwidth of each device's link to the switch. I also examined the captured packets to find if any retransmissions had occurred, but there were none. So no packets were lost even when all devices in the network were communicating with each other continuously. This shows that data transfers in my test environment were *reliable*.

I could also send data to the ASUS router, just as fast as sending data to Comp1. This conclusion can be drawn from the ping results by comparing the ping results for Comp1 → Comp2 and Comp1 → Router. This comparison shows that the data rate is not noticeably lower when going to the router, even if there were a noticeable increase in delay. One must consider that these measurements were made in a home network and the performance of the ASUS router is not representative of the general performance of high-performance router and switches (such as those used in a commercial network or data center).

My results are based on consumer grade Ethernet hardware. Their performance may not be optimal. By using industrial grade Ethernet devices applications can have a performance closer to that of the data-link layer which would improve throughput and reduce delay. If others conduct research in this area, industrial grade Ethernet hardware should be evaluated as was done in Muhmmad Ibrahim's thesis project [17]. However, consumer grade hardware is improving in performance and I expect that in the future the differences will be negligible. Future work should attempt to simulate a real automobile with devices representing the different systems existing in an automobile. Such a network should support communication with as many devices as currently exist in a CAN system. In this thesis, I only examined a very small LAN which cannot be compared to the much larger LAN that would be in an automobile.

If I had to redo this thesis project, I would have created a larger test network with several shared switches and routers rather than just a small LAN environment. This would replace the WAN tests and the congested network test while providing test results that could be reproduced by others. There are many unknown variables in the WAN tests and these tests would be nearly impossible for others to reproduce.

I would also suggest comparing different categories of Ethernet cables to to see if they make a difference in performance. These tests should be done in controlled environments such that each cable experiences the same physical stress as all other cables. I would also do Wi-Fi tests to see if some links could be replaced by Wi-Fi without sacrificing reliability in terms of latency and packet losses.

I would also conduct the thesis project at a company that has better hardware than was used in this thesis project. I would compare the better hardware to the customer grade hardware that I used in order to see how well they perform for the tests I ran. It would also be desirable that ISPs provide an overview of their networks and further details of the configuration of the routers and switches in their network I would like to monitor latencies in larger networks and transfer much more data in each tesst in order to have more accurate measurements than were possible with the ping tests I conducted.

The main conclusion from this thesis project is that Ethernet can serve as a replacement for CAN, but there is room to improve Ethernet's reliability and support for time-critical communication. In addition, there is room for more research, especially with regard to jumbo frames and IPv6. If hardware performs better with jumbo frames in terms of both bandwidth and latency then the network will be less congested and the total per packet processing time (such as processing header fields) will be reduced when a large amount of data needs to be sent. More research regarding IPv6 is also needed, as we see that in terms of throughput IPv6 is slightly faster than IPv4, although it has a higher latency.

If the latency could be reduced for IPv6 even when packets are encrypted then IPv6 over Ethernet offers a potential candidate to replace CAN.

One of the difficulties of the tests that I made was measuring bandwidth using TCP traffic rather than UDP traffic. Although TCP provides reliable data transfer the data needs to be received in a specific order and ACKs need to be sent. In the future SCTP should be researched and widely deployed so that it becomes a commonly used protocol, similar to TCP/UDP. In this case, I expect that it would perform at the same levels while avoiding unnecessary traffic in the network. SCTP's multiple streams can be used to avoid the head of the line-blocking problem of TCP.

My final conclusion is that SCTP, jumbo frames, and IPv6 require more research in order to improve their performance in terms of throughput and latency.

6.2 Future work

I have left out SCTP tests because this protocol is relatively new, while the TCP and UDP protocols are far older and therefore more research has been done regarding improving the performance of these two protocols. Operating systems have integrated these two protocols in their kernel. In contrast, native SCTP support is not available for the Microsoft Windows operating system and third-party software is needed to utilize this protocol. SCTP is sometimes not implemented in test programs. Therefore tests of SCTP were not done and hence there is limited information about the performance of SCTP. I believe that it is important to encourage people within the industry measure and improve the performance of SCTP, such that this protocol could serve as a replacement for TCP or UDP.

More research is also needed for jumbo frames. The suggested MTU in the Ethernet standard has led to most devices in the global Internet being configured for this MTU. Jumbo frames are desirable to reduce the computational time for the CPU. Additional research is needed to explore if network efficiency would increase if jumbo frames were more widely used.

Future work should examine the performance that is possible with commercial grade Ethernet switches, such as those made by Westermo Research and Development (http://www.westermo.se/). Although consumer grade hardware provides low latencies and high bandwidth, it is an open question if they could provide sufficient performance when used in an automobile communication system. As noted above, more devices should be connected into the LAN in order to provide results that could be more easily reproduced than my WAN tests.

One should also compare different cable types, such as shielded compared to unshielded cables, to see if this improves the success in delivering packets when the cables are in environments where bit errors are more likely to occur (keeping in mind the large currents that will flow in electric vehicles – for example due to the use of linear motors/generators in place of shock absorbers). This testing should be done in order to ensure the reliability of data being transferred over a cable. Additionally, it will be important to understand what types of cables are the most efficient when it comes to transferring power by using PoE. If more devices can be powered using PoE, then this help eliminate unnecessary cabling and unnecessary weight.

In this thesis I used TCP/IP over Ethernet to conduct my tests, however the TCP/IP stack model is not optimized for time-critical applications. One area for future work is to explore what gains are possible by avoiding the network and transport layer (to be more comparable to CAN). This would bring the software closer to the hardware and potentially improve performance of the network with respect to minimizing delays. Additional research is necessary to see what types of forward error correcting and other coding schemes can be used to make such a network more reliable for time-critical applications.

6.3 Required reflections

This thesis project is important because not only because Ethernet could be an alternative to CAN, but by using the Ethernet's PoE capabilities the power consumption when driving a vehicle can be reduced. This is because CAN system does not have the ability to power attached devices, while Ethernet can power devices using PoE – hence reducing cabling and the weight of the cables. If the weight is reduced this reduces the energy needed to move the vehicle which has a positive impact on the environment. While the reduction in fuel consumption has a positive impact on the environment, it negatively effects the private and global economy since the demand for fuel will be reduced.

A more efficient communication system can also reduce the amount of energy needed to transport a message from one node to another; this in turn reduces the electrical power consumption in automobiles. Even if the differences in some scenarios can be ignored due to the potential energy savings being so small, the cumulative effect cannot be ignored.

This thesis project briefly reviewed the Internet Protocol and noted that IPv6 is the standard to adopt for use in vehicles. This version of IP comes includes IPSec, which can be used to ensure that data transferred between nodes is not altered, changed, or even read by other network devices among the path. This feature can help avoid personal integrity problems that could arise with Ethernet based communication system; it can also be used to protect a car from any interference from the outside by corrupt hardware or programs. This feature is very important since that Ethernet is a widely used technology and many people work with it, hence if someone wants to hurt a person they could attach a malicious device to fool the rest of the system with data that could hurt the user. However, by encrypting all data - the system can ensure that the source IP address was not spoofed nor the data modified.

References

[1] B. Ranjan and M. Gupta, “Automobile Control System using Controller Area Network,” *Int. J. Comput. Appl.*, vol. 67, no. 18, pp. 34–38, Apr. 2013.

[2] C. Watterson, “Controller Area Network (CAN) Implementation Guide,” *Appl. Note -1123 Analog Devices Inc*, 2012.

[3] “Cut-Through and Store-and-Forward Ethernet Switching for Low-Latency Environments - Cisco.” [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-465436.html . [Accessed: 11-May-2014].

[4] “Understanding Switch Latency - Cisco.” [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-3000-series-switches/white_paper_c11-661939.html . [Accessed: 11-May-2014].

[5] James F. Kurose and Keith W. Ross, *Computer Networking A Top-Down Approach*, 5th ed., vol. 2009. Pearson, 2009.

[6] J. Postel, “Internet Protocol,” *INTERNET PROTOCOL. Internet Request for Comments*, RFC 791, Sep. 1981 [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt [Accessed: 14-Jul-2014].

[7] Stephen E. Deering and Robert M. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” *Internet Protocol, Version 6 (IPv6) Specification, Internet Request for Comments*, RFC 2460 (Draft Standard), Dec. 1998 [Online]. Available: http://www.rfc-editor.org/rfc/rfc2460.txt

[8] J. Postel, “Internet Control Message Protocol.” *Internet Request for Comments*, RFC 792, Sep. 1981 [Online]. Available: http://www.rfc-editor.org/rfc/rfc792.txt [Accessed: 14-Jul-2014].

[9] J. Postel, “User Datagram Protocol.” *Internet Request for Comments*, RFC 768,, Aug. 1980 [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt [Accessed: 12-Mar-2014].

[10] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson, “RFC 2960 - Stream Control Transmission Protocol.” RFC 2960, Oct. 2000 [Online]. Available: http://www.rfc-editor.org/rfc/rfc2960.txt [Accessed: 08-Apr-2014].

[11] Randall R. Stewart, *Stream control transmission protocol (SCTP) : A reference guide*. Boston, Mass: Addison-Wesley, 2001.

[12] J. Postel, “Transmission Control Protocol.” *Internet Request for Comments*, RFC 793, Sep. 1981 [Online]. Available: https://tools.ietf.org/html/rfc793. [Accessed: 12-Mar-2014].

[13] R. T. McLaughlin, “CAN Overview,” in *1997/384), IEE Colloquium on CANopen Implementation, Digest No*, 1997, pp. 1/1–1/27.

[14] A. R. Kanthla, *Network Performance Improvement for Cloud Computing using Jumbo Frames*. Master's thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, TRITA-ICT-EX-2014:27, March 2014. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-143806

[15] S. Narayan and P. R. Lutui, “TCP/IP Jumbo Frames Network Performance Evaluation on A Test-bed Infrastructure,” *Int. J. Wirel. Microw. Technol. IJWMT*, vol. 2, no. 6, p. 29, 2012.

[16] Konrad Etschberger, “Comparing CAN- and Ethernet-based Communication.” [Online]. Available: http://www.ixxat.com/download/artikel_comparison_can_and_ethernet.pdf . [Accessed: 14-Jul-2014].

[17] M. Ibrahim, *Ethernet in Steer-by-wire Applications*. Master's thesis, KTH Royal Institute of Technology, School of Information and Communication Technology,TRITA-ICT-EX-2011:175, July 2011. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-91049

[18] J. Sandberg, *Inter-Vehicle Communication with Platooning*. Kandidate thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, TRITA-ICT-EX-2014:96, July 2014. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-147916

TRITA-ICT-EX-2014:108