University of Twente

Faculty EE-Math-CS
Department of Electrical Engineering

Real-time network performance characterization

Locbus v.s. RTnet

E.Buit

Pre-doc Assignment

Supervisors prof.dr.ir. J. van Amerongen
dr.ir. J.F. Broenink
dipl.ing. B. Orlic

may 2004

Report 011CE2004
Control Engineering
Department of Electrical Engineering
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands

Summary

The aim of this project is to get insight how Real-time Ethernet behaves with respect to the field bus called Locbus. By writing a benchmarks for both systems the characterisations of both buses is performed. The transmission is time, the time which necessary to get the data to and of the card. Variability of measured data or so-called jitter is also determined.

From the research it becomes clear that Ethernet is faster than Locbus. Locbus however is assumed more stable because the protocol is generated in the hardware. The test for this statement could not be performed because interrupts are not supported by the Locbus hardware. For large quantities of data Real-time Ethernet is recommended using, because then the speed will play a larger role.

To get a better insight if it is realisable for a field bus to be replaced by Real-time Ethernet it is advised to build a test set-up which uses Real-time Ethernet.

Samenvatting

Het doel van dit project is om inzicht te krijgen hoe Real-time Ethernet zich gedraagt ten opzichte van de veldbus genaamd Locbus. Door voor beide systemen een benchmark te schrijven is gekeken naar de karakteristieken van beide bussen. Hierbij zijn de transmissie tijd, de tijd die nodig om de data naar en van de kaart te krijgen en de zogenaamde jitter gemeten.

Uit het onderzoek blijkt dat ethernet sneller is dan Locbus. Locbus echter wordt stabieler geacht omdat het protocol in de hardware wordt gegenereerd. De test hiervoor was niet mogelijk omdat interrupts niet worden ondersteund door de Locbus hardware. Voor grote hoeveelheden data is het aan te bevelen Ethernet te gebruiken omdat juist dan de snelheid een grotere rol gaat spelen.

Om een beter inzicht te krijgen of het realiseerbaar is om een veldbus te vervangen door Real-time ethernet is het aan te raden een opstelling te bouwen die gebruik maakt van Real-time ethernet.

Contents

1 Introduction

From the 80's fieldbusses replaced wiring in parallel of input and output (IO) to and from controller cabinets. Using fieldbus is a cheaper and easier way of connecting IO, because only one cable has to be connected from the cabinet to the IO instead of multiple. Because speeds of production lines are increasing fieldbusses can not transport the amounts of data to steer all the IO that is required to drive the modern complex machine. Also distributed control comes into the sight making the demands on data transportation even higher.
It is possible to connect multiple systems by a fieldbus, but this is a rather slow and expensive way of doing. Real-time Ethernet can be used to do the job. This is much cheaper, because commonly used hardware can be used. If used right, real-time Ethernet can give even higher throughput rates at the same stability fieldbusses can, but some aspects have to be kept in mind. This report is about the characteristics of real-time Ethernet compared to a fieldbus called Locbus.

1.1 Context of the project

The major concern of University research projects in the area of fieldbuses is to compare various fieldbuses from the viewpoint of real-time behaviour and resistance to transmission faults. This project has also practical significance, because it is conducted in the context of Stork Plastics Machinery (SPM) as industrial partner.

At SPM in Hengelo, plastic extrusion machines are being built. These machines are equipped with the ARTOS [Locamation] real-time operating system and a serial bus called Locbus [Locamation]. Both these products are made and maintained by Locamation [Locamation], SPM tends to make shift towards using standard components. Achieving better performance while decreasing costs is also an issue. Another point is that the machines are very expensive, while the profit is relatively low. Scalability is also a reason to look at other ways of doing because the expectation is that their system will be too slow to keep up with the ever growing market demands in the (near) future.

The first step made by SPM was to try another RTOS. Orientation towards using standard components has led them to choose Linux extension specialized for real-time called RTAI [Rtai]. This setup could meet the expectations and SPM has decided to review if similar replacement with standard component can also be done in a case of network.

A general system uses a controller to control the plant and get the user input. The controller is connected by a serial and a VGA cable to a user interface and by a fieldbus connection to a remote IO block. From the IO rack all sensors and actuators are wired in parallel. The general setup is given in figure 1.

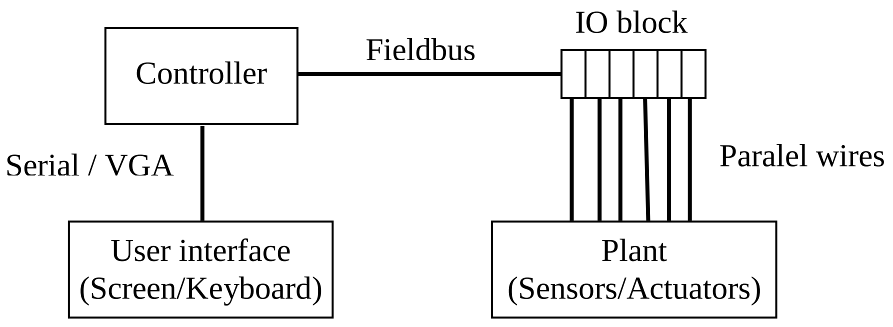


Figure 1: General setup

Note that Controller block shown in Figure 1 might be implemented either using one centralised computer or using distributed system of computers.

The university proposed to look into the system of SPM and see if they can find a way to improve the system for the future and to get an industrial example of the academic versions now used. While SPM is using fieldbuses merely as a way to connect I/Os to a central computer, the University research group is more concerned with implementing distributed systems. However there is a common interest of comparing performance of some standard components.

1.2 Outline of the report

This report will describe the theory, the results and the conclusion of this project.

Chapter 2 is concerned with initial choices that have been made regarding bus protocols, operating system and underlying hardware.

In chapter 3 the backgrounds on Locbus, RTnet and RTAI will be discussed. A comparison between Locbus and RTnet will be given at the end of chapter 3.

In chapter 4 the benchmark used to determine the characteristics will be discussed for both RTnet and Locbus.

In chapter 5 are the results of this project and in chapter 6 a discussion and conclusion will be given. Recommendations on how these results can be used in future projects will also be given in chapter 6.

2 Setup space exploration

2.1 Introduction

There is more than one way to build a setup with the same functionality. Fieldbus protocol, operating system and hardware platform are parameters that can be changeed. There is also a choice whether a system is implemented as a centralised or as a distributed one. In this chapter spectre of options that might be interesting for SPM and/or University research group will be given. SPM is interested in both characterising performance of components they currently use (ARTOS, Locbus) and comparing them to performance of some more standard components (RTAI, RTnet, CAN…)

To choose between different options an review of all the options will be given and associated pro’s and con’s will be analyzed. In the block diagram figures depicting various options of implementing general setup from Figure 1, block representing the plant is left out for clarity reasons..

2.2 Options

2.2.1 Option 1: Testing the ARTOS setup as used at SPM

The idea is to use the same setup, as it is used by SPM, and get it up and running in our lab and do some performance tests. After these tests, from the results conclusions can be drawn whether this setup is too slow for the future.

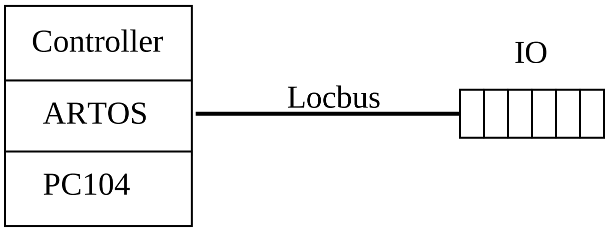


Figure 2: ARTOS/Locbus setup

Pro:

- In this way a good reference can be obtained of what the current setup is able to do.
- An industrial setup that looks and feels like the real thing is available on the university.

Con:

- Since Locbus is not likely to become a standard fieldbus, University research group is not too much interested in using it after the test are finished. Thus buying the PC104 Locbus master and the IO is not an option.
- Fact that ARTOS is not an open source system makes it more difficult to invent accurate tests.

2.2.2 Option 2: Testing the RTAI setup as used at SPM

The idea is to also to use the setup, as it is used by SPM, but now with RTAI operating system instead of ARTOS. The Locbus driver for RTAI already exist [Engelen], but no real testing has been done on the RTAI system. After these tests, the bottleneck of the system can be determined, and a new option can be designed and tested.

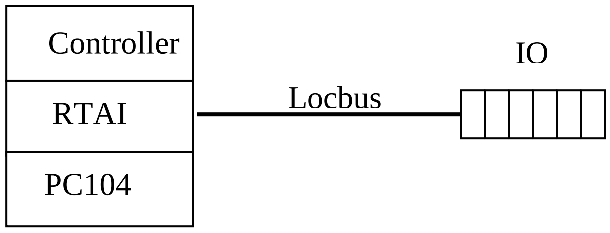


Figure 3: RTAI/Locbus setup

Pro:

- In this way a good reference of what the RTAI setup is able to do is obtained.
- An industrial setup that looks and feels like the real thing is available on the university.

Con:

- Same as for option 1, once after the test are finished the PC104 Locbus master and the IO are not very useful in the University lab.
- The current driver is made to support the intermediate RTAI/ARTOS system [Engelen]. This means that the Locbus driver is not really a driver but an application specific module. If a comparison should be made to know how Locbus is compared to other busses like CAN, Interbus or Profibus all drivers should be implemented in same way. In case of RTAI this could mean transforming the driver according to the LXRT standard of RTAI (EXTENDING-LXRT). The advantage of LXRT will be discussed later on.

2.2.3 Option 3: Using a distributed setup based on RTnet

Compared to previous option in this one instead of the Locbus fieldbus RTnet is used. RTnet is a protocol based on User Datagram Protocol (UDP) service of the Internet Protocol (IP) to guarantee real time behaviour. To avoid unpredictable collisions on the Ethernet, a dedicated Ethernet segment is required.

Also instead of centralised system, two PC104s are used: one as a user interface and partly as controller (master node) and another PC104 as IO handler and also partly as controller (slave node). IO cabinets can be attached to both nodes. .

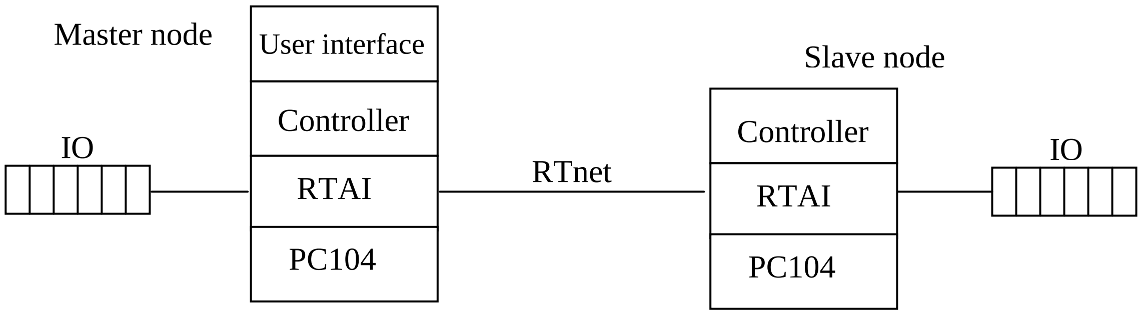


Figure 4: RTnet setup

Pro:

- University research group is interested to build distributed systems based on the library (CT library) developed at this group in order to use concurrency in structured and scalable way in distributed real-time systems. This setup fits nice into such a view of the project, because this is truly distributed system. Besides real-time network protocols based on cheap Ethernet components are already ongoing trend in the industry.

- This research group is already involved in other PC104 based projects and described setup can, after the experiments are performed, be combined with other PC104 projects.

- Processing power can be better used because of decentralisation. If the Master/Slave paradigm is applied, the amount of data exchanged between Master and Slave nodes can be much less, due to the fact that the Slave node has certain processing power as well. Unlike hardware implementation of Slave node as merely I/O cabinet, the software implementation offers more flexibility in the way data is packed. For example design might specify that from Master/ to Slave only command parameters are sent and in opposite direction only the data that is to be logged and/or displayed is send.

- 100 MB/s Ethernet can transport large amounts of data faster than 2MB/s Locbus.
- Almost all of the modern PC104 boards have an Ethernet connection. The test setup can be of standard hardware from the consumer market so that the initial costs can be low. This is also significant advantage from economical point of view.
- The IO cards on the control system can be of any price and specs, so more flexibility.

Con:

- Two computers are needed for the same job.
- The part of the program implementing the supervisory control and operator display needs to be separated from the part implementing the controller.

2.2.4 Option 4: Porting the Hilscher driver to RTAI.

To accurately compare the different busses, it is necessary to use a system where the bus performance is the only changing parameter. To realise this, the underlying hardware and software should be the same.

Hilscher is a manufacturer that makes hardware for most of the leading industrial serial busses [Hilscher]. Several different fieldbus setups would be used in this option. Although I/O controllers and peripherals are from same manufacturer and designed in similar way, for each type of bus specific hardware is needed. However there is only one driver that can handle up to 4 of these cards, no matter what kind. This driver is at the moment only available for Linux without real time extensions.

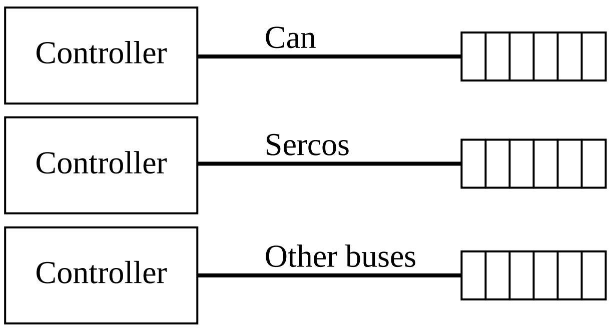


Figure 5: Hilscher setup

Pro:

- When the driver is ported, all the busses are available for testing.

Con:

- Dependency on Hilscher

- Expensive because for each bus type another card and configuration software have to be bought.

- The card has its own processor that needs handshaking with the main CPU. This takes some time and generates jitter. This degrades the performance significant and the transport latency will get to high.

2.3 Chosen solution and setup

	Industrialiseability*	Future	UT interest
Option 1 ARTOS	++	--	--
Option 2 RTAI	++	-	-
Option 3 RTnet	--	++	++
Option 4 Hilscher	+-	+	+-

* Industrialiseability is ability to use the setup in robust way in industrial setting…

Conclusions:

- SPM is interested in comparison of currently used components and possible alternatives. From University point of view, investing money in equipment that will be used only for one time measurement is not economically justified. Therefore joint decision was made to borrow some Locbus IO and a PC104 Locbus card from SPM in order to obtain measurment data on their current system.

- The third option is the preferable one for the UT because the tendency of the industry is to use of Real-Time Ethernet as a fieldbus. It can also be used for many applications and especially because the CT libraries can be tested on a distributed system based on Ethernet. This option is also interesting alternative solution for SPM. The risk with the RTnet option is that the industry will not accept the results because of industrialiseability. If performance measurements and cost analysis show significant advantages of options 3 and 4 SPM might decide to apply them instead of option 1 or 2.

- The last option is excluded because of earlier test done by SPM [Stork].

The chosen way was to invent benchmark tests to measure both the characteristics of Locbus and of the RTnet. The test results will be compared and from those tests it should be possible to be concluded whether RTnet can replace Locbus and in what environment.

3 Background info on Locbus, RTnet and RTAI

3.1 Locbus

As the result of another student's project [Engelen] supervised by UT and SPM, the original SPM's driver for the Locbus card on ARTOS was,ported to RTAI. .. However, that driver was made in order to communicate with SPM's control software and was therefore still application specific in some aspects. To make successful comparison with other drivers, all drivers must be on same general level of functionality. Thus in this project some of the application specific functionality had to be removed from driver, while other features were needed that where not yet in the existing driver.

3.1.1 Interrupts

A feature that is not used is the interrupt. The manual of Locamation states that interrupts are supported by the card. However on the board that we have used to perform experiments interrupts are not supported because of a hardware failure in the Locbus master. In order to see when the data has arrived, polling of the card is needed to see if data is available. The Locbus hardware is really sensitive on time moments of polling: if polling is done to fast, faster than the physical time of the round trip time, the data gets lost in the polling process. In order to avoid loss of data, the roundtrip time of the Locbus has to be calculated and polling needs to be done at that rate. If polling is done at the calculated rate, no data is lost.

3.1.2 Transport latency

The speed of the bus is 2 MBit/s so one bit can be send in 500 ns. A complete roundtrip the time per bit can also be considered to be 500ns because the data is echoed in full duplex. The typical packet is structured like this [Locbus]:

Flag	Address	Function	Data	CRC	Flag

- Flag: a one byte Flag that is automatically generated by the hardware.
- Address: a one byte address of the slave to send the data to. The maximum number of nodes is 255.
- Function: a one byte Function value.
- Data: contains the actual data that has to be send. The maximum amount of data is determined by the structure of the system. Each node can contain up to 12 racks. These racks may contain all kinds of IO, eg. 48 digital IO, 16 analogues IO or a 14 channel high speed temperature card.
- CRC: two bytes of the cyclic redundancy check (CRC)
- Flag: a one byte Flag that is automatically generated by the hardware

Thus for every packet sent, 6 bytes of protocol overhead are added. For every byte of data another byte of protocol overhead needs to be send. The roundtrip is almost equal to the transmission time because the stream is echoed with a one bit delay for each node, so for every node we have to add 0,5 μs. The time of the header and the trailer takes 24 μs to send.

For every byte of data 8 μs must be added, 4 μs for the data and 4μs because for the data overhead. Figure 6 shows the path of the data in the Locbus ring.
Time in μs in formula form:

$$T_{rtt} = (6 + 2B) \cdot 4 + 0,5 \cdot N = 24 + 8 \cdot B + 0,5 \cdot N$$

B = number of sent bytes
N = nodes

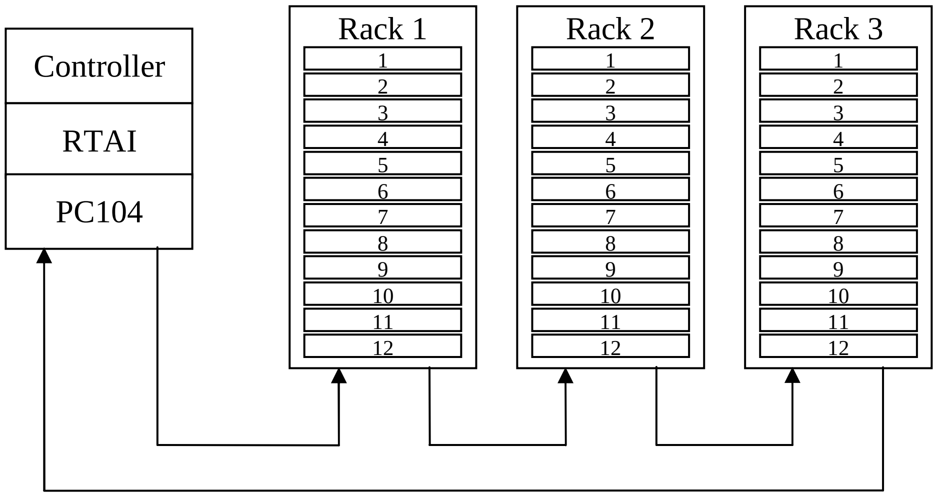


Figure 6 Locbus setup with 3 racks, 12 nodes each

3.2 RTnet

RTnet has been developed at the University of Hannover [Rtnet]. It uses the User Datagram Protocol (UDP) service of the Internet Protocol (IP) to guarantee real time behaviour. If the Transmission Control Protocol (TCP) was used, the overhead and data correction mechanisms would have prevented corruption and/or loss of data, but would also decrease the potential for the real time usage.

3.2.1 Configuration and bus access protocols of RTnet

In a normal environment Address Resolution Protocol (ARP) is used to translate an IP address into MAC address. Because ARP is not yet supported by RTnet, a different kind of initialisation is needed to start up the network. The Real-Time Configuration Service (RTcfg) is designed to do precisely that. RTcfg takes care of exchanging Media Access Control (MAC) addresses and optionally IP addresses. After the client is configured to be a member of the network, it can start sending and receiving data.

If more than 2 nodes are in the system or an asynchronous media access is used, an arbiter system is needed to control the media access. In RTnet Time, Division Multiple Access (TDMA) is standard implemented, but others can be used. In TDMA bus access scheme every member of the network may access it to send real-time data only in a predefined time slots. In this way real time data can be send in a deterministic way, and also non real-time data can be send in the slack time of the slot. In RTnet there is one server node and one or more client nodes. At the beginning of every time frame the server node sends a

Start Of Frame (SOF). All the client nodes know offset of their time slots to the SOF and send the data only when they are allowed to.

3.2.2 Transport latency

With Ethernet there is some overhead because the packet needs to be addressed, even if a point-to-point connection is used. The contents of an Ethernet packet are shown below:

Preamble	Ethernet header	IP header	UDP header	data	CRC	Bit gap

- Preamble consists of 8 bytes for synchronization of the network
- Ethernet header consists of 14 bytes containing the source and destination MAC address
- IP header consists of 20 bytes of all sorts of IP information including the source and destination IP address
- UDP header consists of 8 bytes containing the source and destination port
- Data is data to be transported and can be of variable length with a maximum of 1472 bytes.
- CRC consists of 4 bytes containing the CRC32 value of the complete packet
- Bit gap must be at least 12 bytes long to signal the network the transfer is complete

Since the network is running at 100 megabit per second, the time needed to send one bit is 10 ns. If the roundtrip time is calculated of an empty packet the total number of bytes that has to be transported is 66 bytes. Sending these 66 bytes takes 5.28 μs. If doing a roundtrip, the time needed to transport the data back and forth and is thus twice the transportation time. The time for the roundtrip becomes 10.56 μs. For every byte of data that is send in the roundtrip another 160 ns has to be added to the time, because 8 bits need to be send back and forth.

Time in μs in formula form:

$$T_{rtt} = (66 + B) \cdot 0.16$$
$$B = bytes\ to\ send$$

3.3 Theoretical comparison of Locbus & RTnet

Locbus and RTnet are hard to compare because Locbus is a real fieldbus and RTnet is a multipurpose bus. If one wants to use RTnet as a fieldbus the structure of the data has to be defined by the user. With locbus this structure is defined by the connected nodes.

The packet overhead of RTnet is much larger than for Locbus as can be seen in Figure 7.

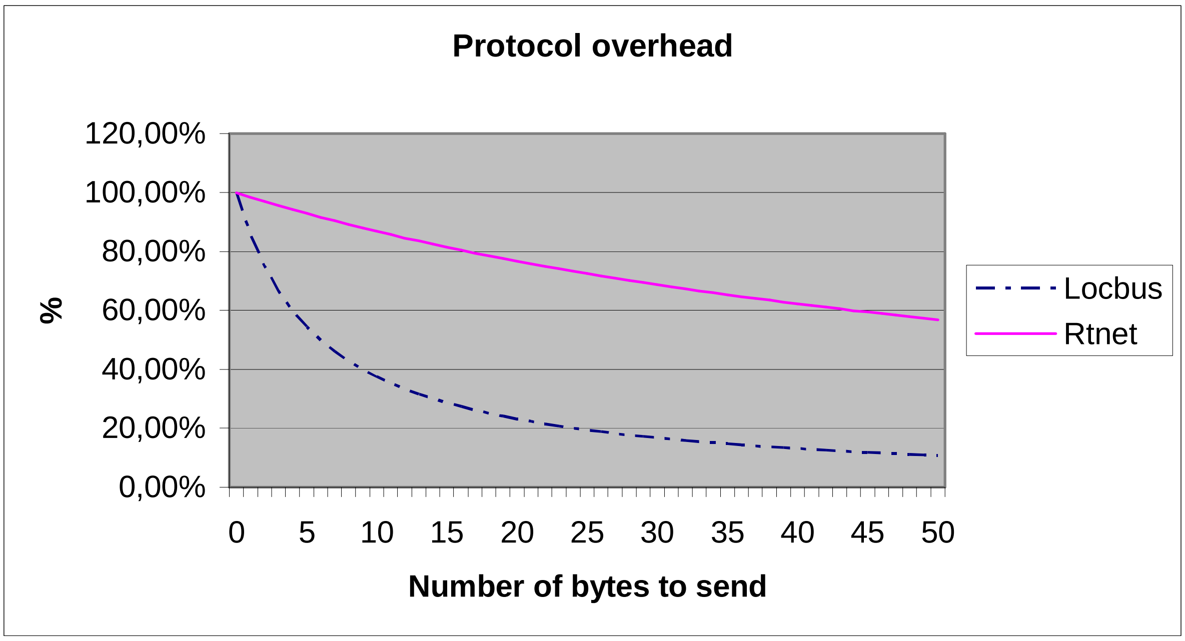


Figure 7: Protocol overhead of Locbus and RTnet

If the physical latencies of the two are compared, as is done in Figure 8, it is obvious that physical roundtrip time of RTnet is much less than for Locbus despite the larger overhead. The reason for the difference is the much higher transfer speed of RTnet.

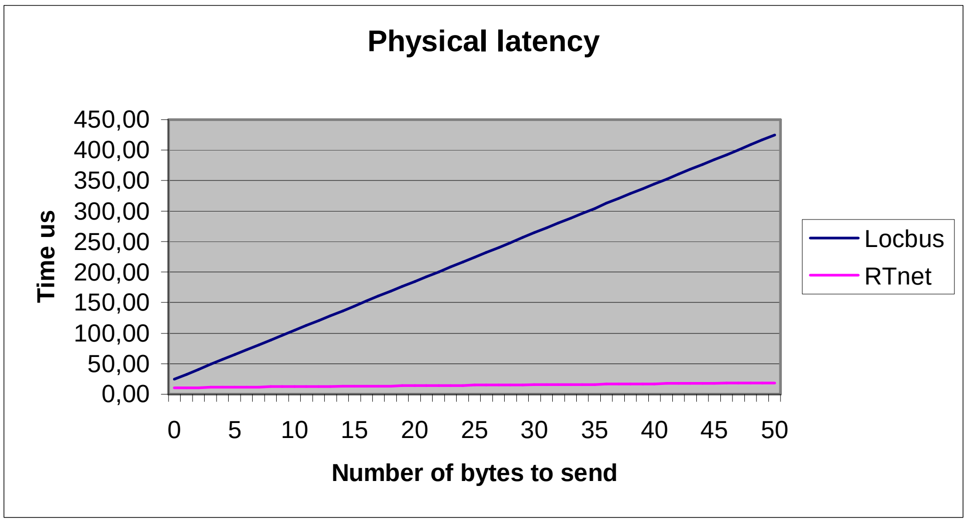


Figure 8: Physical latency of Locbus and RTnet

3.4 RTAI

Real-time Linux Application Interface (RTAI) is a real-time extension of the Linux kernel invented by Paolo Mantegazza at the University of Milan [Rtai]. RTAI was developed in order to allow deterministic responses to events, as is needed in real-time systems, while keeping all the benefits of existing Linux operating system.

Normally drivers are written in kernel space and if they meet certain criteria they can be accessed from user space. Linux modules which are not drivers and are working in kernel space cannot be accessed from user space. A major advantage of user space is that if something goes wrong, the rest of the system will continue to work normally. In kernel space however, if something goes wrong, it really goes wrong. In the worst scenario an OOPS failure is generated and the system has to be rebooted in order to continue. It is therefore strongly advised to access drivers from user-space.

In order to do real time reading and writing from user space, RTAI delivers a module called LXRT. This module makes communication from user space to kernel modules possible. Because a real-time hardware driver is just a kernel module, it is possible to communicate to hardware from user space with LXRT.

A disadvantage of LXRT is that context switching is needed to go from user space to kernel space and vice versa. The time needed for this action is in the order of tens of microseconds. In industrial practice if performance is really an issue the application developed as user module after proven it is stable enough can be moved to kernel space to prevent switching.

4 Drivers & Benchmark

In order to compare the Locbus with real-time Ethernet a benchmark is needed. It would be nice if a benchmark is developed that can be use to quantisize future busses.
What has to be determined by the benchmark is:

- Read/Write time
- Roundtrip time
- Stability or so called jitter

The first thing that has to been known for a good comparison is the time the CPU needs in order to put the data from the memory onto the bus(write time) and to get the data from the bus into the memory (read time).

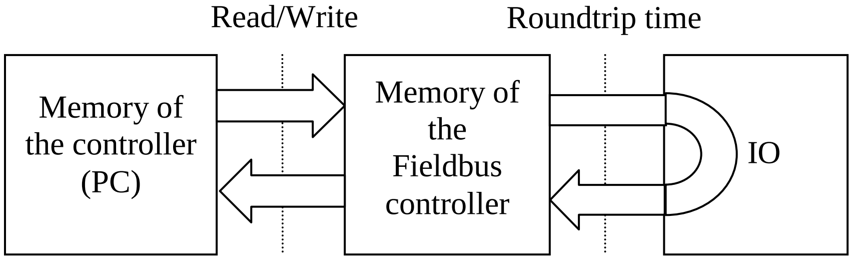
The second thing that is important for the characteristics of the bus is the roundtrip time. Roundtrip time is the time elapsed starting from the moment when writing a message for the field device has started till the moment data is returned back. Normally in the control systems field devices only generate output from received data or only send measured data to the controller. Although peripheral devices in control systems usually do not echo received data back to the controller, this is very common measure in benchmarking fieldbusses because time can be measured on same node eliminating the need to synchronize clocks. However note that sending the data in upward and downward direction might not be symmetrical in some networks.

The third thing that is of major importance is the stability in the measured values of write, read and roundtrip time, the so called jitter.

The first thing that has to be measured, the time to get the data onto the board, can be accomplished by getting the time when the driver returns from its data transfer action and subtracting from that the time when the data transfer action has started.

The roundtrip time can be measured by sending data to the device and waiting for it to return its data, if then the start time is subtracted from the arrival time the roundtrip time is calculated. To measure the roundtrip time one can use several starting times. In this benchmark it is best to use the time the write action of the data has been finished, in order to get a full cycle. To do a precise measurement, the interrupt has to be enabled to get the quickest response to the incoming data. If interrupts are not available and polling have to be used instead, it is much more difficult to perform precise measurements.

The stability is the last parameter that needs to be known. A third variable is kept that store the average time. Figure 9 shows the complete roundtrip.



Read/Write
Roundtrip time
Memory of the controller (PC)
Memory of the Fieldbus controller
IO

Figure 9: Data flow during one roundtrip time

4.1 Implementation of the Locbus benchmark

For the Locbus benchmark the driver had to be rewritten to be used in a different way. With the SPM version of the driver, the data needs to be formatted to the SPM format of the data. At the university the only concern is how to get the data in on and off the card. So new `read()` and `write()` functions had to be written. The method used in the new driver is to fill a buffer with the data that has to be send and give a pointer to this buffer to the driver. The driver takes care of the rest like formatting and placing the right data on the right place. When reading data from the Locbus, a pointer to a buffer that is allocated to receive the data from the Locbus is given to the driver. The driver fills the buffer with the received data in a way it is easy to use at the user side.

4.2 Implementation of the RTnet benchmark

A universal test in kernel space has been build that can determine the network characteristics of the network types supported by RTnet on all kinds of hardware. The benchmark is in the client server model. Because software latencies are a much bigger problem with RTnet than with Locbus, due to the faster response time of Ethernet hardware, all the modules will run in kernel space to prevent switching.

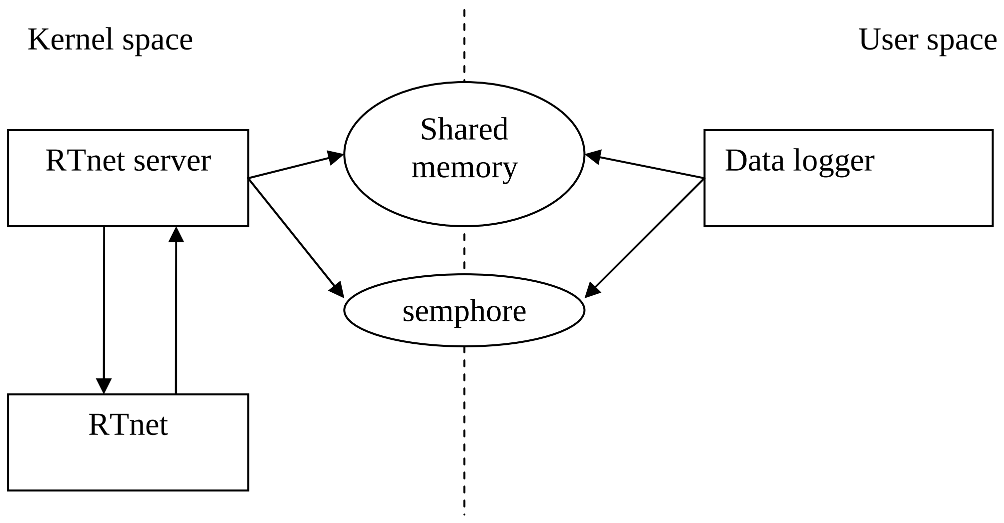


Figure 10: RTnet benchmark server module

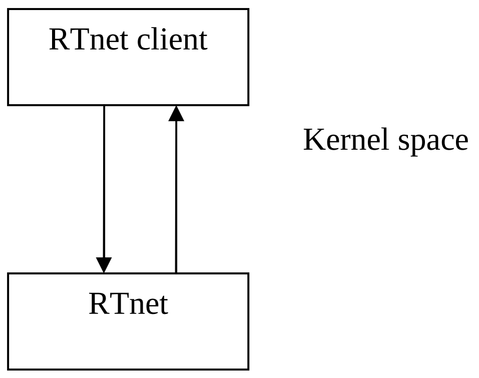


Figure 11: RTnet benchmark client module

First the server will be discussed. The server is build up of two components, as shown on Figure 10. A kernel space module will handle the data from the RTnet and store the measurements to a shared memory. The second component - data logger is a user space program that reads the shared memory and stores the data to the disk. The user space program is needed because from kernel space, file access is not permitted. The semaphore is used to signal the data logger that the measurement is finished and the data can be written to file.

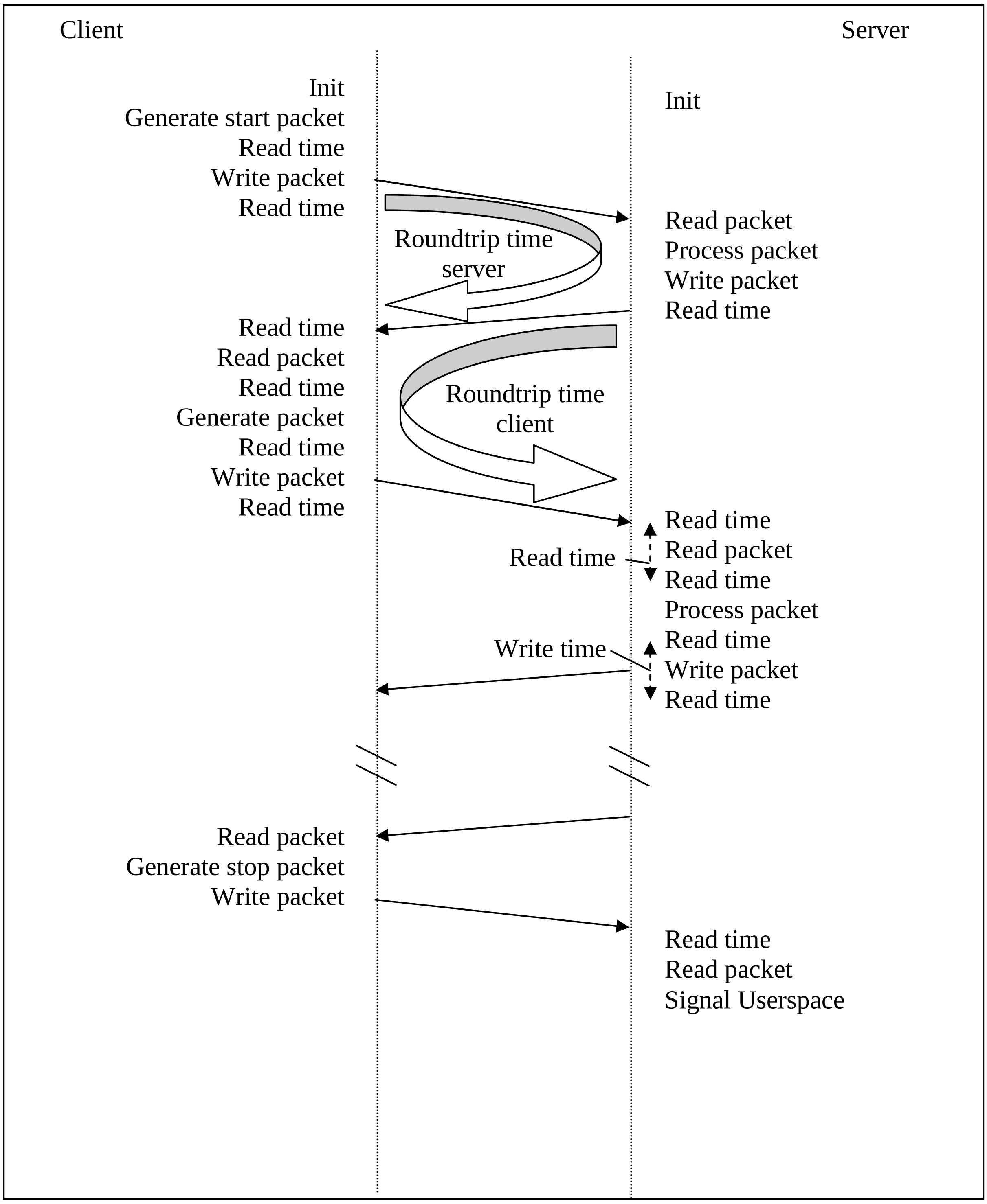


Figure 12: Benchmark sequence diagram

The client only consists of one kernel space module that communicates only with RTnet as shown on Figure 11.

The benchmark works as shown on Figure 12. After both the client and server have started, the client will send an initial packet to the server. This packet contains information about the test, like number of cycles and the type of the client. The start and stop time of the write call are stored to calculate the write time. The time when the client returns from its write call is stored to calculate the server roundtrip time.

The server receives the packet, processes it and sends back an echo of the message. The time the server returns from its write routine is stored to determine the client roundtrip time.

When the client receives the echo it will store the start and stop time of the read routine to calculate the read time and the roundtrip time. A packet is now constructed that contains the read, write and roundtrip time. Because the times are send to the server the clients side does need to have a logger and all the measured data are on the server side. The packet containing the measurements is send to the server. The start and finish time of the write call are again measured for the next packet.

When the server receives the packet it will get the time the packet arrives and the time that is needed to get the packet. An echo of the message is returned to the client. After the message has been send, the read, write and roundtrip time on the server side are calculated. These values along with the ones received from the client are stored in the shared memory.

This loop will continue until the client reaches its number of cycles and will now generate a stop packet. When the stop packet is received by the server it will signal the user space program that the benchmark is ready. The user space program now reads the memory and stores the contents to a file.

The start up of real time Ethernet and the benchmark can be performed executing a script that does the necessary work. See the Appendix I for the README of the benchmark on how to use it and what options can be used.

4.3 Hardware used

For the Locbus test a Geode 300 MHz with a PI300 PC104 Locbus master card is used. As node a LC210 48 digital IO card borrowed from SPM was installed to act as node.

For the RTnet tests the following four systems* have been used.

- 3,5 " SBC**: Geode 300 MHz with a Realtek 8100 NIC
- PC104: Via Eden 600 MHz with a Realtek 8100 NIC
- Desktop PC: AMD Athlon 1000 MHz with an Intel EtherExpress pro NIC
- Laptop: AMD XP Mobile 1800 MHz with an National Semiconductor DP8381x NIC

*All network cards support 100 MBit Ethernet and the test are all done at this speed.
**SBC stands for single board computer, which is often of the 3,5"size (floppy disk drive), 5,25"(PC size CD player) and PC104 are also form factors.

Both RTnet and Locbus measurements are done with 50.000 packets generated at the server side.

5 Results

In this chapter results of performed measurements are summarized. Unfortunatelly, the fact that event based communication (using interrupts) was not possible for Locbus, has prevented fully compatible comparison between Locbus and RTnet. However, RTnet benchmark was performed for all combinations of the four hardware node types and has produced very interesting results.

5.1 Locbus

Because with the Locbus most of the communication is handled in the hardware, the times needed to send and receive a packet are very small and do not have much variation. The average time of one roundtrip is 115 µs. Writing to the board takes about 18 µs, this is the switching time between user and kernel space. Reading time is much harder to determine because the interrupts are not supported. If 1 ms is taken as wait time, to be sure that the data must be there, the reading time takes the same time as the writing, this again because of switching between kernel and user space. If performed in kernel space, the reading and writing time of this particular test would drop to 4 µs because only 4 registers have to be written. Reading would take longer because whole Locbus packet has to be read, because else it wouldn’t be possible to tell if the packet is correct or corrupt.

Because of interrupts not being supported by the Locbus hardware another measure of time is needed because the results are not accurate. When calculating the time needed for the roundtrip by Locbus and do measurements in software, one sees that if polling faster than the calculated time packets are lost. If polling at the rate of the calculated roundtrip time or slower one sees that no packets are lost and the expected results come up. It can thus be assumed that the theoretical time of the packet roundtrip time is the real roundtrip time.

5.2 RTnet

RTnet communication times are mostly determined by the hardware. Not only processor speed but also architecture and type of network interface card (NIC) are of great importance. In appendix II, the average and the standard deviation of the read, write and roundtrip time are given. On the Y axis are the number of times the value is measured, on the X axis is the roundtrip time.

In appendix III the measured time values are given. The Y axis is the time in ns and X axis contains the packet number. On the left hand side are the times as they are measured by the server. On the right hand side are the times as they are measured by the client. The upper graph is the read time, the middle the write time and the lower graph is the roundtrip time measured by that device. This is the physical latency of the Ethernet plus the time the other side needed to process the packet.

The left figure of Figure 12 shows the write time of the PC in the PC->Laptop experiment. The left figure shows the read time of the PC104 in the PC104->Laptop experiment. Let us here remind that Geode processor runs at 300MHz, PC104 runs at the frequency of 600MHz, PC at the frequency of 1000MHz, and laptop at the frequency of 1800MHz.

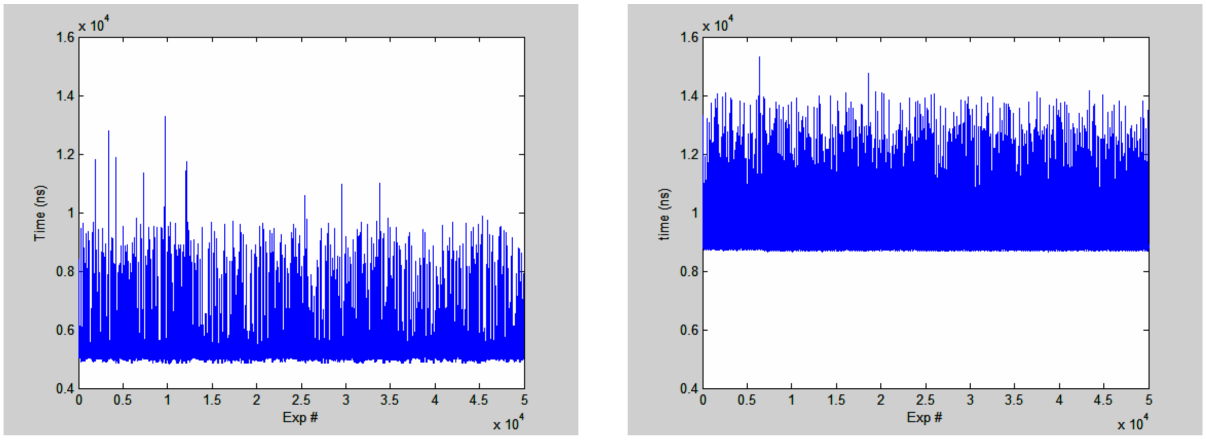


Figure 13 Writing by PC(left) and PC104(right)

What can be concluded from Figure 13 is that writing in the case of the PC104 is more stable than in the case of PC. The average of the PC write time is 5161 ns (Appendix II) and the average of the PC104 is 9192 ns. When looking at the worst case peaks, they are, compared to the average values, proportionally much bigger and more often in the case of PC than in the case of PC104. This indicates that PC104 is more deterministic in writing. Standard deviations, which are 404 ns on the PC and 667 ns on the PC104, or proportionally 7,8% of the average value for the PC and 7,2% for the PC104, also indicate that PC104 is more deterministic.

The left figure of Figure 14 shows the read time of the geode in the PC->Geode experiment. The right side shows the read time of the laptop in the PC->Laptop experiment.

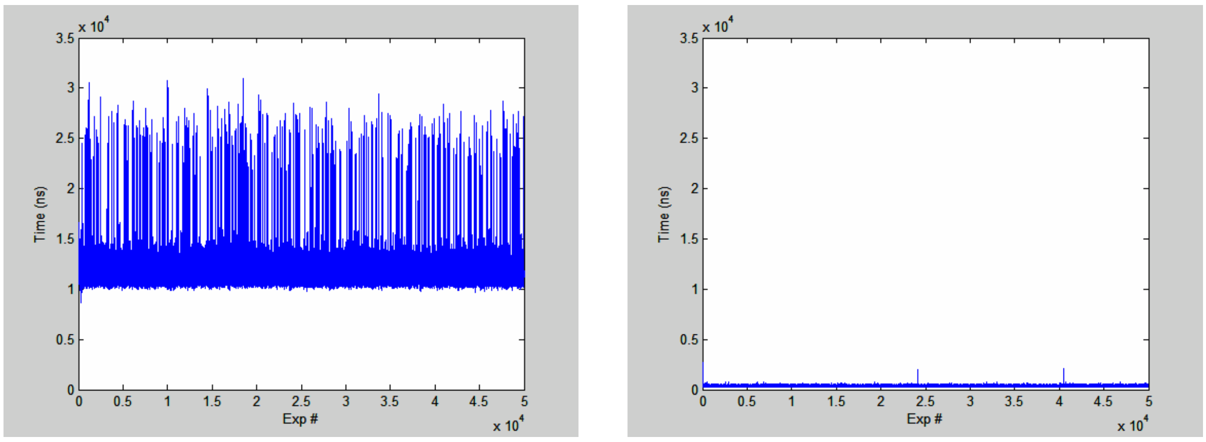


Figure 14: Reading by Geode(left) and Laptop(right)

What can be seen is that read time of the laptop is much faster, average of 452 ns for the laptop against 11,9 µs for the Geode. A difference could be expected because the processor speed is 6 times higher. What also can be seen is that the jitter of the Geode much larger compared to the laptop. The Geode has lots of peaks away from the average while the Laptop has only a few peaks. Still those few worst-case peaks of Laptop are proportionally very large compared to the average read time.

All the graphs of the experiments can be found in Appendix III. It has to be noticed that unlike in presented test examples, the graphs in Appendix III are not always on the same scale. This is done in order to give a better view of the deviation and effects.

Figure 15 shows the roundtrip time as function of the processor speed. What can be seen is that an increase in processor speed in general will give less roundtrip time, because a lot of work is done by the software.

For each node type two type of roundtrip time measurements exist: one in which that node is server and other one in which it is a client. Thus for each node, two curves will exist depicting its characteristics- one when it is a server, and the other when it is a client. In both cases three measurements are possible: one with each of other types of nodes on other end. Exception is PC104 for which it was possible to use two exactly the same nodes. Because of this PC104 curves has also measured roundtrip time for case in which other node used is also PC104.

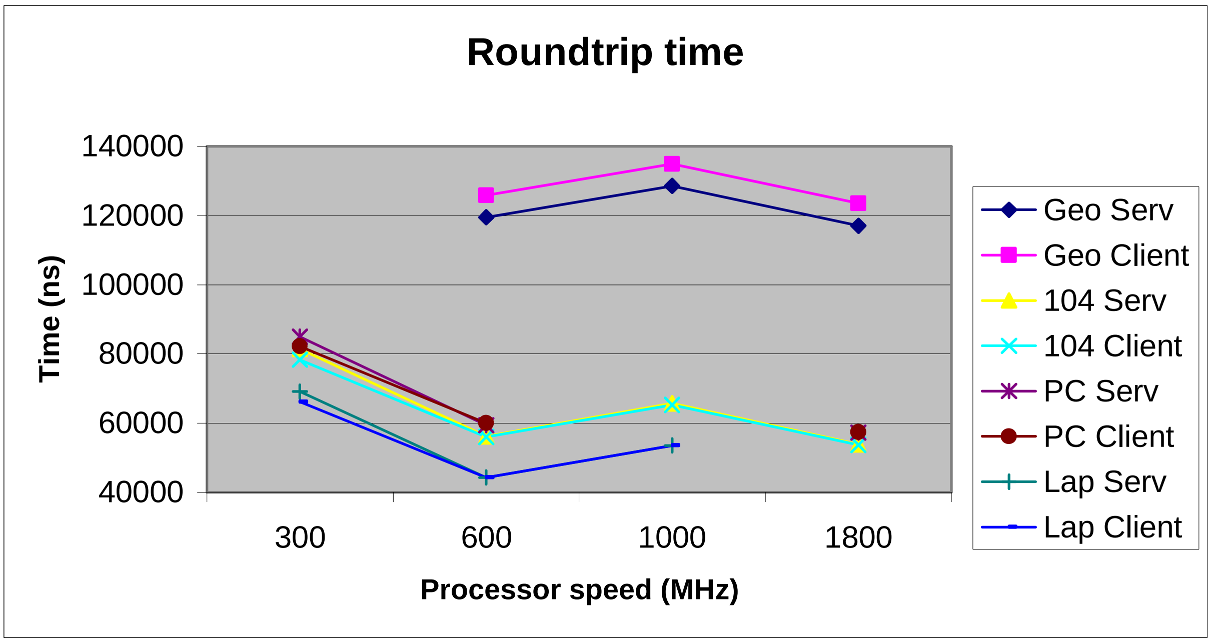


Figure 15: Roundtrip time as function of processor speed

What can be noticed on Figure 15, if looking at the PC104 and the PC points, is that an increase in processor speed does not need to mean that the roundtrip time will decrease with the same amount or decrease at all. The processor speed is increased by 400 MHz but the average roundtrip time is increased instead of decreased. This proves that the roundtrip time of RTnet is dependent on more than just processor speed and that setup using PC104, although slower in processor speed, is faster than the PC. What also can be noticed is that with higher processor speeds the difference in code of server and client does not matter anymore in the roundtrip time measurments.

Figure 16 shows the same figure without the Geode to have a better overview.

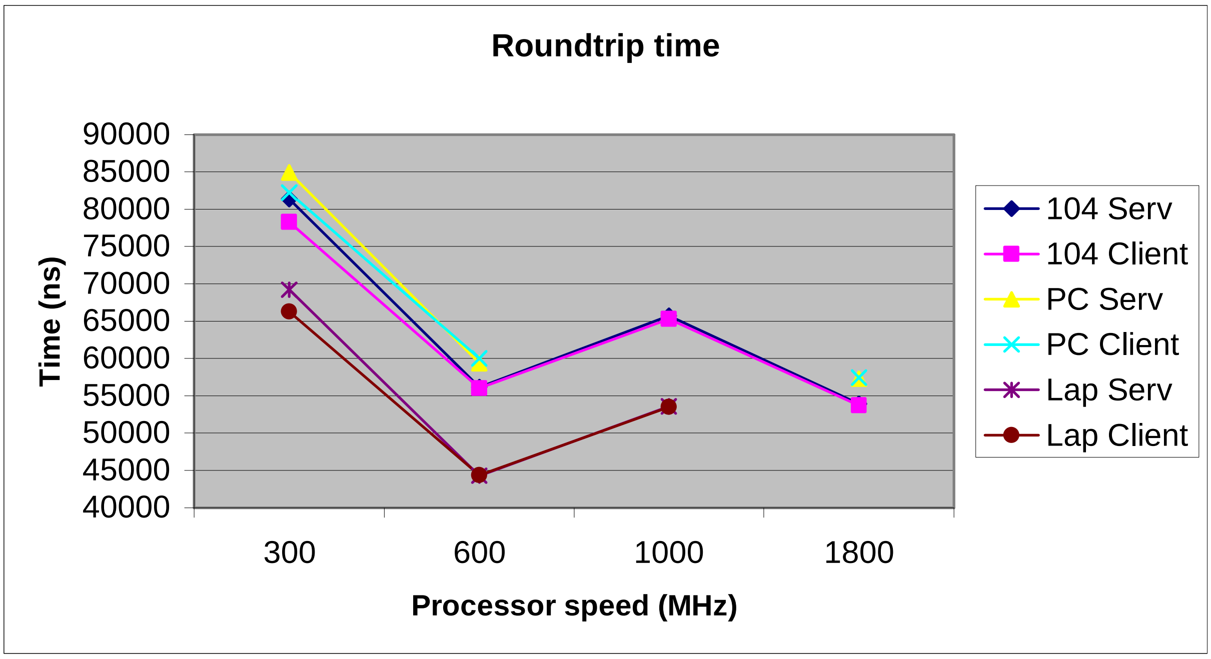


Figure 16: Roundtrip time as function of processor speed

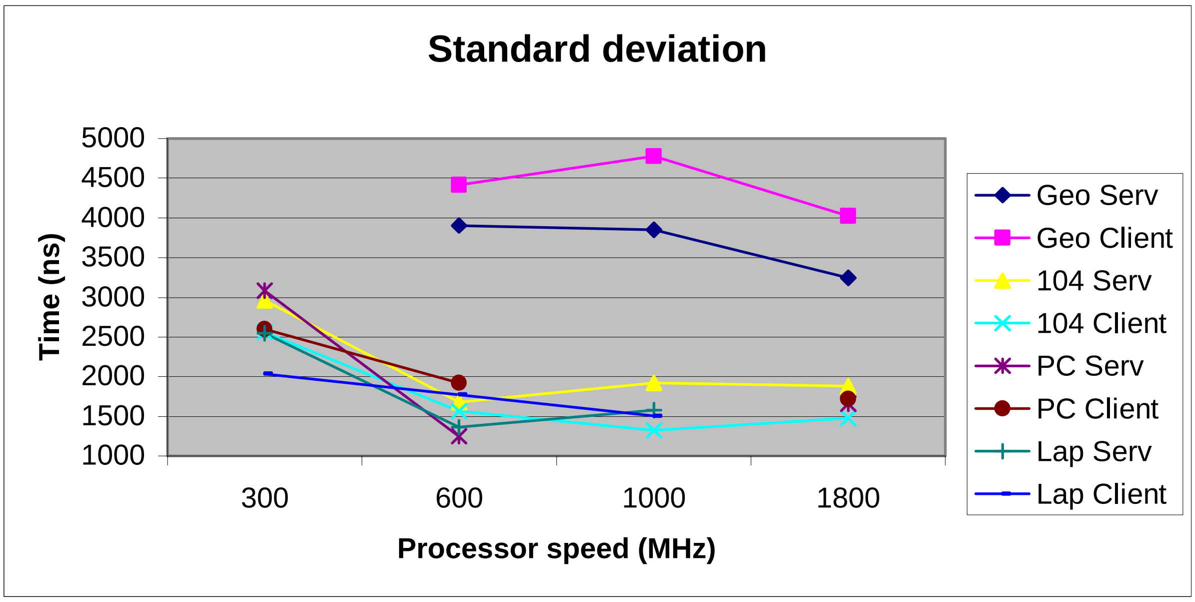


Figure 17: Standard deviation of the roundtrip time

Figure 17 shows the standard deviation of the roundtrip time as measured on a device with a certain processor speed. What can be noticed is that the Geode does not perform well compared to the other three. What also can be noticed is that the deviation gets smaller as the processor speed of both the server and the client increase. Figure 18 shows the same graph but again without the Geode. What can be noticed is that the difference in server and client do appear in the standard deviation.

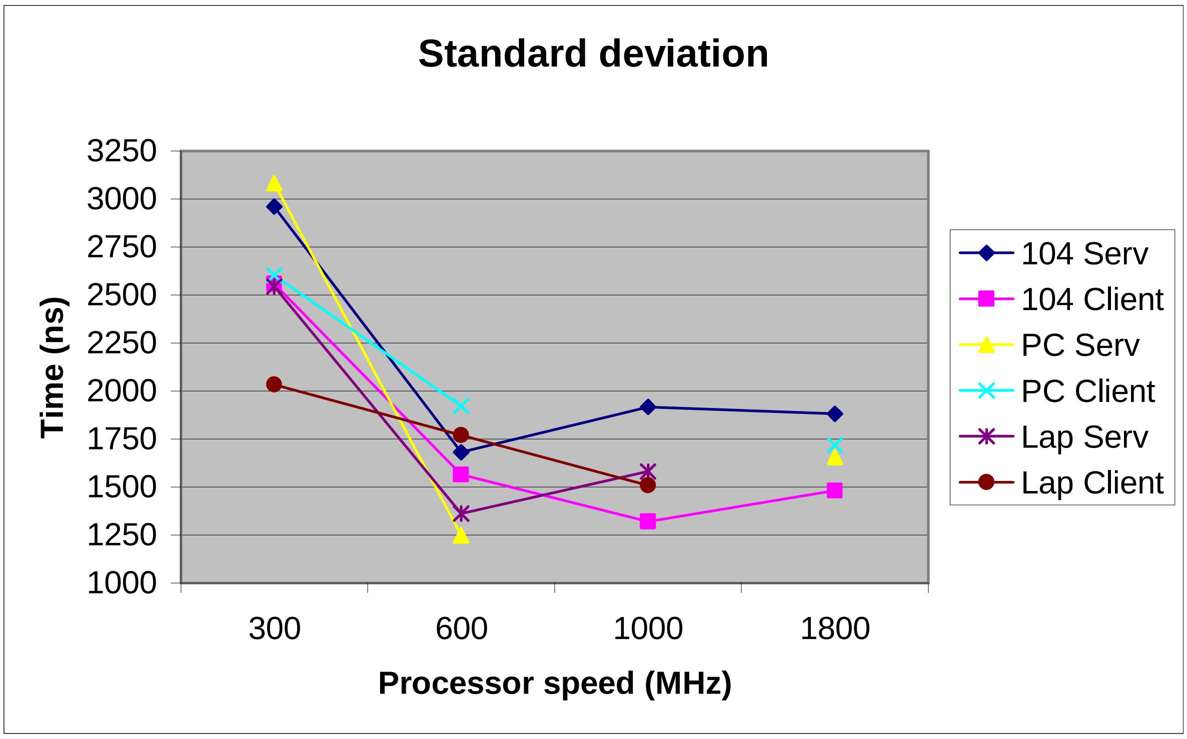


Figure 18: Standard deviation of the roundtrip time

6 Conclusions

6.1 Discussion

Locbus is a very stable and fast field bus. It also takes low CPU consumption because the locbus protocol is mostly generated in the hardware. With Locbus the theoretical roundtrip time of 70 µs with 48 digital IO can be accomplished. For every byte of data two bytes must be added to the packet. This means that for every byte of data, sixteen times 500 ns has to be added to the roundtrip time. This makes the bus rather slow when large packets are being sent.

What can be noticed immediately, when looking at the RTnet results, is that the Geode can not compete with the other three systems. As can be seen in Appendix IV the distribution of the roundtrip time is very large with the Geode. This means that the system is not very stable, which is the first priority when controlling real-time processes.

If the Geode is excluded, it can be said that RTnet is also very stable but much faster than Locbus. In the experiments with the PC104, 58 bytes of data can be sent in the roundtrip in less than 60 µs. The average standard deviation is 1900 ns. If these times are compared with the Locbus it can be seen that 10 times more data can be send in 10 µs less time. If the size of data is increased the resulting differences will even get bigger.

What might be is of more importance is how much CPU power is needed to send this data. With Locbus1 µs of time is used for every two bytes that are written. The same time is needed for reading, but the packet overhead forces to read more data than is necessary.
The PC104, which is the most industrial of the three, uses 1380 ns for reading of 58 bytes from RTnet and writing to RTnet takes 9210 ns. The total time becomes 11 µs for reading and writing the data. In 11 µs Locbus can transfer 22 bytes of data to or from the board because one read or write action to the ISA bus can transfer 2 bytes in 1 µs.

What is excluded from these numbers is the interrupt handler and the hardware and software latency. The average roundtrip time of the PC104 is about 60 µs.

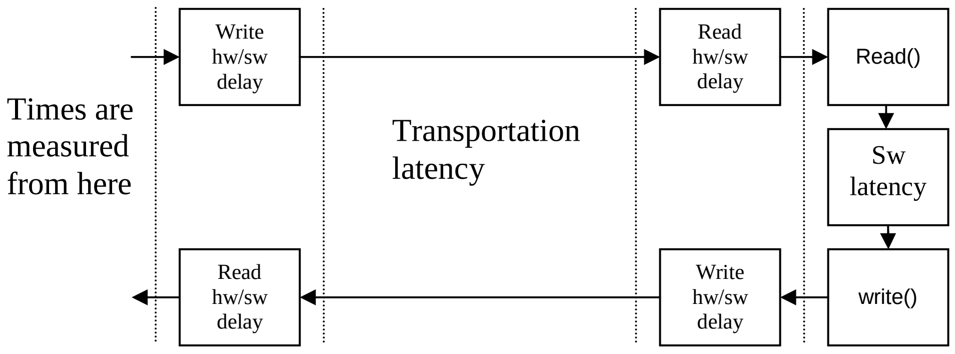


Figure 19 Model of latencies

The complete loop is given in figure 19. The transportation latency, which is 10.56 µs plus 58 times 160 ns, becomes 19.84 µs. Adding to this the read() and write() routine the total time, as measured, is about 32 µs. The time that is unaccounted for is 30 µs. This is thus the time the hardware, the interrupt handler and the kernel module need to create the echo. The total software time for generating a 58 byte packet plus reading and writing of that packet becomes 40 µs. When this time is compared to the time that was needed Locbus was used, that would be 58 time 1 μs (reading and writing of two bytes) becomes 58 μs. It is clear, that for this size of packet, RTnet can transfer data faster than Locbus can.

6.2 Conclusions

1. RTnet can replace Locbus if used with the right hardware.
2. When using RTnet, other problems, like synchronization, arise from the fact that two controllers are used, but these problems can be solved using the CT library.
3. RTnet can be used as a very stable way of transferring data.
4. With larger amounts of data, RTnet will be much faster then any industrial bus now on the market.

6.3 Recommendations

- Implement RTnet into the CT libraries.
- Do more extensive tests on hardware to see where the bottleneck is.
- Build a set-up that uses RTnet to control plant to show that it works.

Appendix I

```
README file of the benchmark

**** RTnet Benchmark ******
The purpose of this measurement is to determine the latency in the
Ethernet hardware and determine the roundtrip time.


Building:

Chance the include directories in the Makefile of the server and client
directory and type make in both directories.

Using:

It is assumed that the rtai 24.1.13 and RTnet 6.2 are installed and you
are in root mode. Other versions of rtai and rtnet might work but are
not tested. Before using the benchmark some rtai modules have to be
loaded. If these modules are not already loaded the rtai_start script
can be used.

Before the benchmark can be used the config file (rtnet_bench.conf) has to
modified to the right settings.

The measurement can be started from the root directory of the benchmark.

Running the measurement as

client: rtnet_bench client <server-ip> [options...]
server: rtnet_bench server <client-ip> [options...]

server_options:
	datadir=<file-to-store-output>
server_options:
	cycles=<number-of-test-cycles>

After the measurement the server system will contain a
file in the directory <datadir> that is called
<serverkind><clientkind>.txt
The default directory to store the file is /tmp
```

Appendix II

Numerical values of measurements

SW: sever write time	CW: client write time
SR: sever read time	CR: client read time
SRT: sever roundtrip time	CRT: client roundtrip time

Lap->Geo means that that laptop was running as server and the Geode was running as client.

104->Lap	SW	SR	SRT	CW	CR	CRT
Average	9192,478	1377,373	44338,59	1888,975	469,2344	53928,23
Std deviation	666,7559	63,24302	1770,256	180,666	110,1685	1881,46

104->Geo	SW	SR	SRT	CW	CR	CRT
Average	9268,324	1380,264	125835,9	34770,28	12243,01	81380,11
Std deviation	870,9818	62,11674	4412,349	2476,028	1176,731	2960,161

104->PC	SW	SR	SRT	CW	CR	CRT
Average	9208,451	1376,602	59989,67	5190,507	692,7603	65727,38
Std deviation	680,5722	56,04185	1920,932	347,2254	68,56832	1917,855

Lap->104	SW	SR	SRT	CW	CR	CRT
Average	1912,476	472,4951	53748,25	9195,094	1228,353	44295,7
Std deviation	38,79149	91,86295	1479,652	663,9338	80,73615	1360,175

Lap->Geo	SW	SR	SRT	CW	CR	CRT
Average	2034,137	490,8066	123551,3	34881,74	12579,94	69192,88
Std deviation	58,74043	80,76765	4026,291	2096,772	1091,33	2544,216

Lap->PC	SW	SR	SRT	CW	CR	CRT
Average	1851,028	486,1724	57446,45	5174,314	688,5769	53598,99
Std deviation	46,43687	60,66979	1716,574	398,6822	36,61868	1580,268

PC->Geo	SW	SR	SRT	CW	CR	CRT
Average	5123,247	695,6596	134875,9	34814,32	11876,05	84938,88
Std deviation	532,2634	54,00714	4778,097	2107,066	1404,969	3084,703

PC->Lap	SW	SR	SRT	CW	CR	CRT
Average	5161,782	700,0765	53501,77	1872,388	451,6068	57292,4
Std deviation	404,559	63,27627	1507,089	52,69557	62,04034	1655,686

PC->104	SW	SR	SRT	CW	CR	CRT
Average	5038,448	697,568	65283,77	9329,93	1228,152	59352,76
Std deviation	222,811	47,22437	1320,77	734,1399	67,10694	1247,175

Geo->PC	SW	SR	SRT	CW	CR	CRT
Average	33017,79	13407,52	82221,18	5213,937	690,0521	128665,1
Std deviation	1728,337	1007,961	2602,836	468,6242	64,19357	3852,774

Geo->104	SW	SR	SRT	CW	CR	CRT
Average	32808,34	13977,66	78320,87	9318,034	1229,24	119573,4
Std deviation	1823,366	1114,977	2556,973	920,2092	75,76216	3905,078

Geo->Lap	SW	SR	SRT	CW	CR	CRT
Average	33656,75	13082,59	66270,4	1909,128	481,298	117052,2
Std deviation	1747,89	941,2218	2034,29	69,12515	28,70627	3245,455

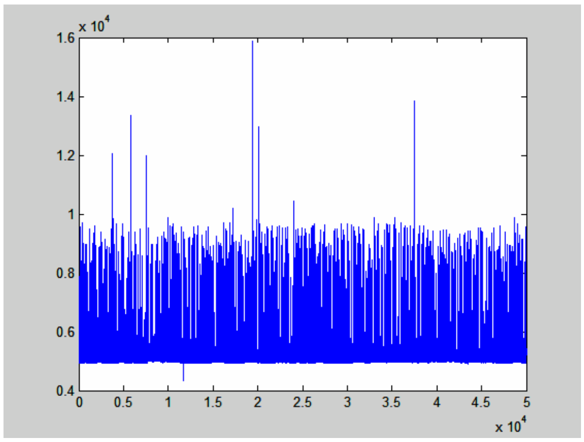
104->104	SW	SR	SRT	CW	CR	CRT
Average	9294,209	1373,857	55993,62	9220,36	1227,955	56172,02
Std deviation	912,6145	54,15547	1564,134	840,5529	59,70907	1679,334

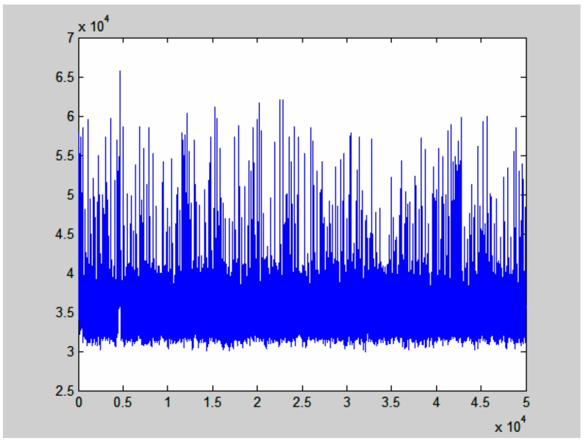
Appendix III

Benchmark measurements

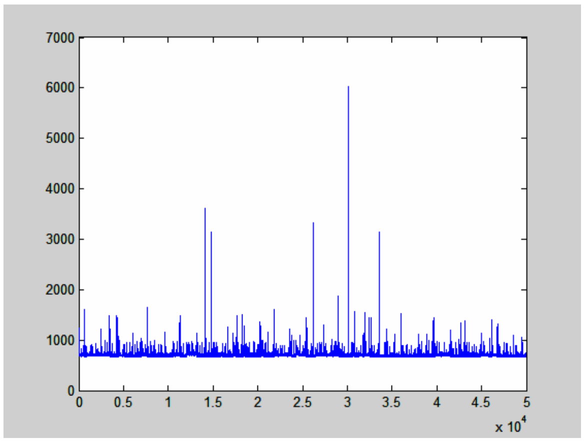
Server:AMD1000 Client: Geode

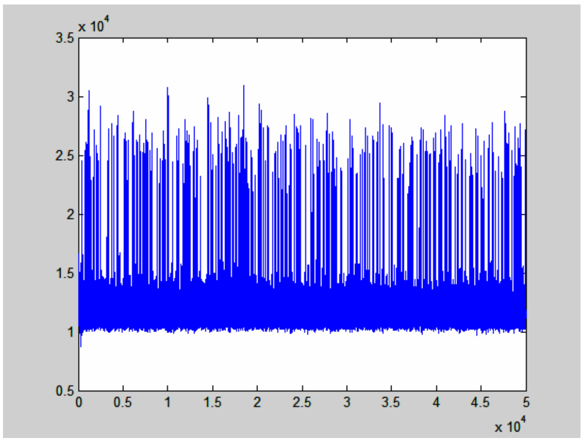
Write



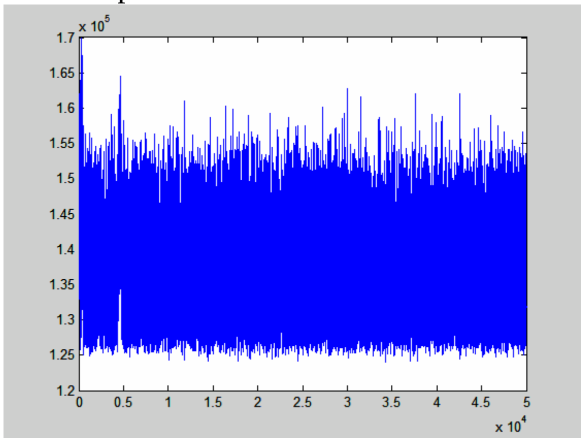


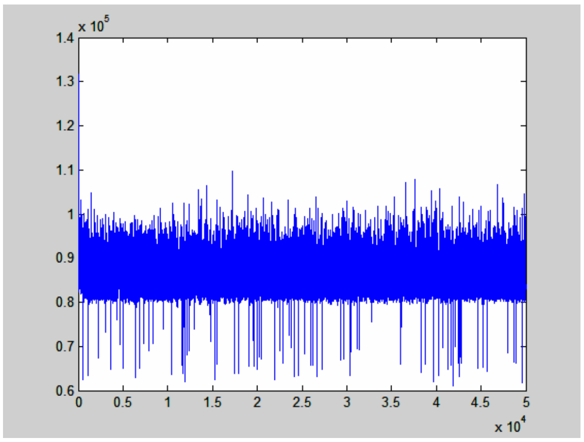
Read





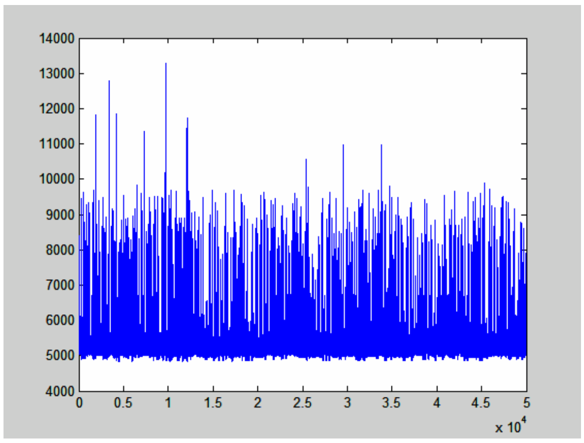
Roundtrip

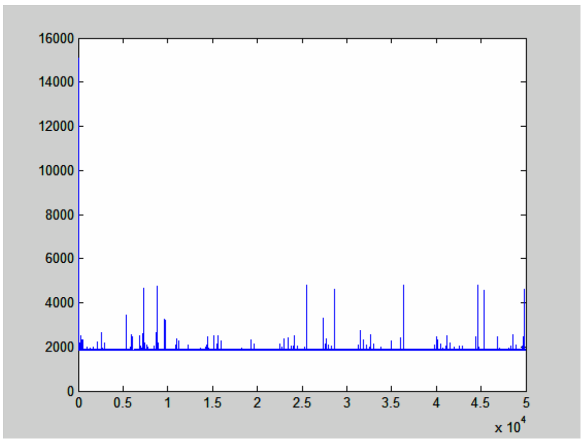




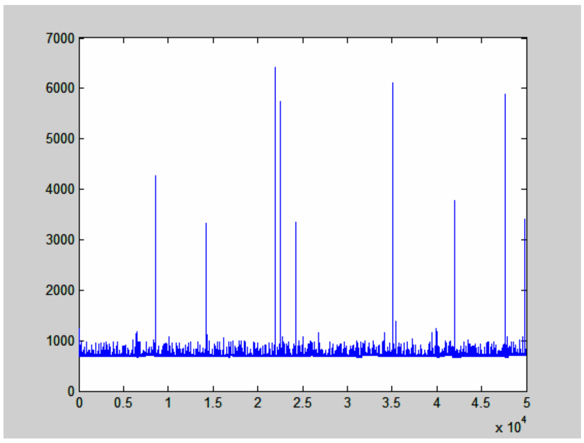
AMD1000 laptop

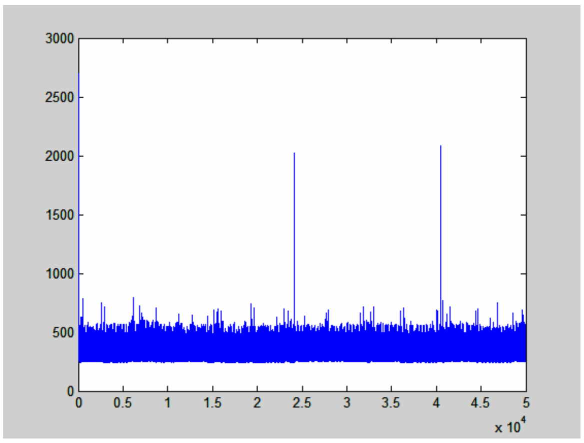
Write



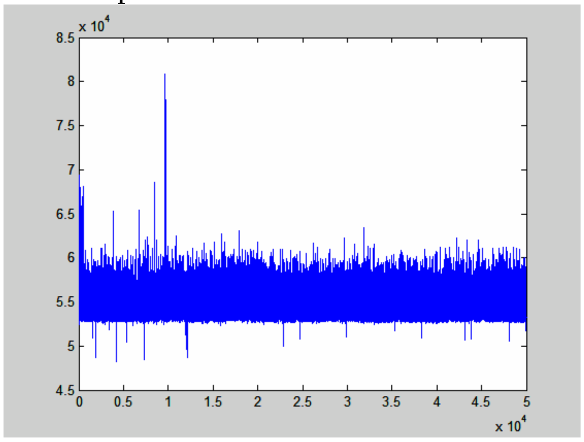


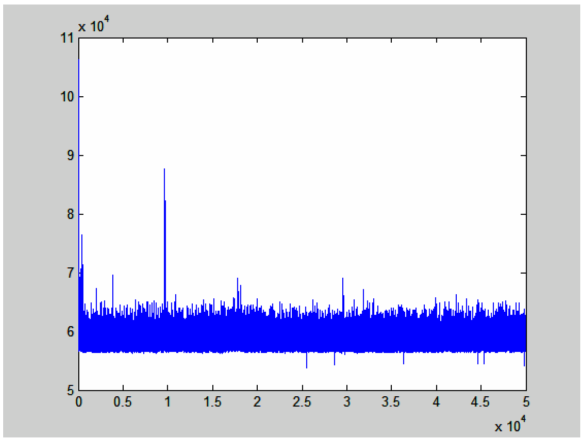
Read





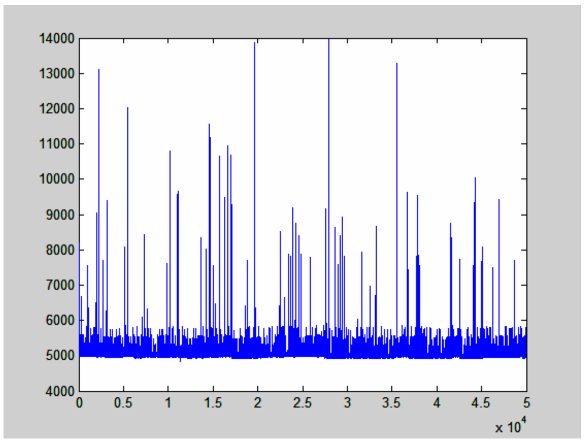
Roundtrip

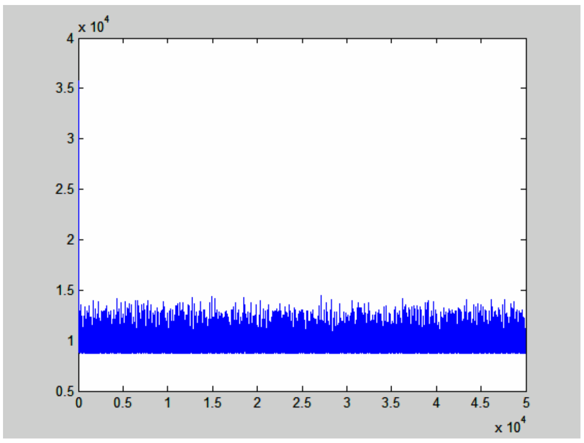




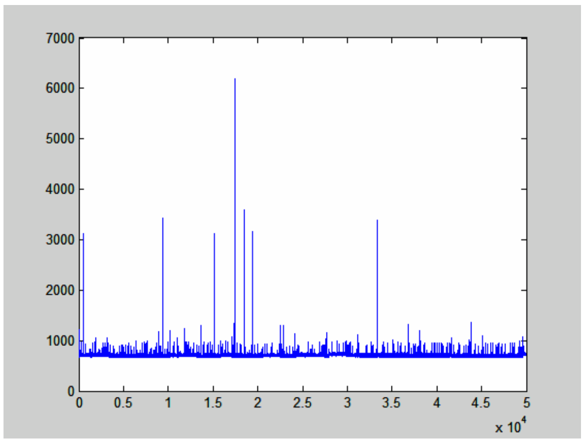
AMD1000 PC104

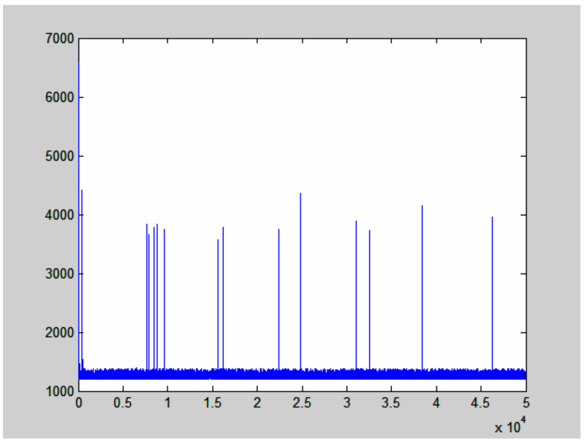
Write



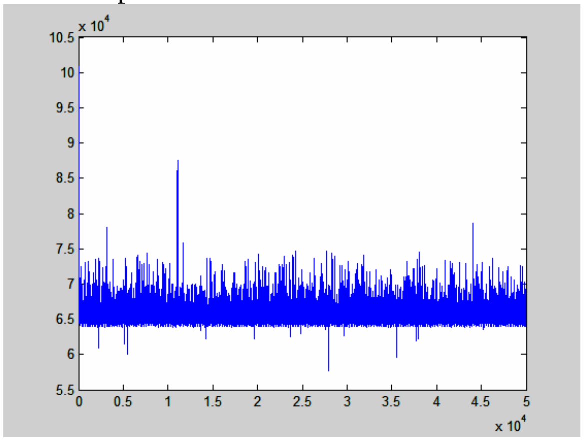


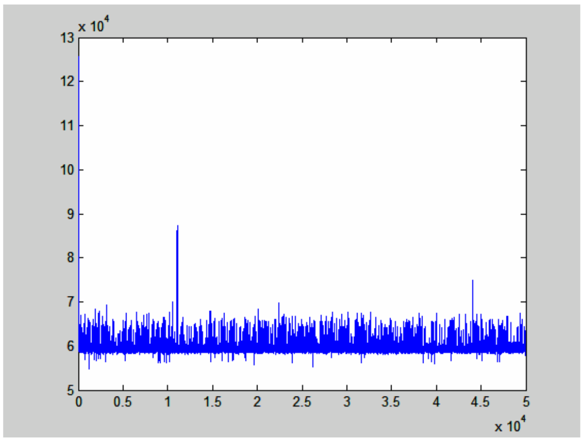
Read





Roundtrip

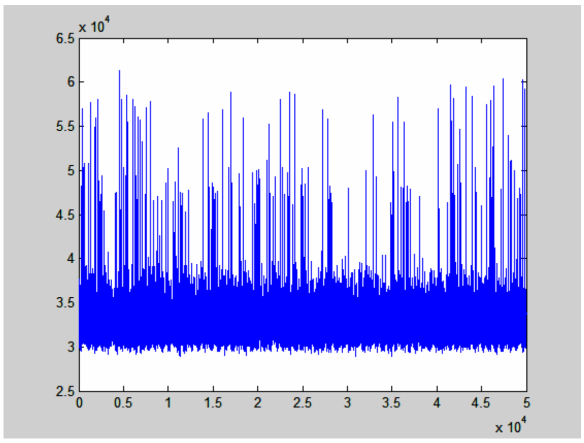


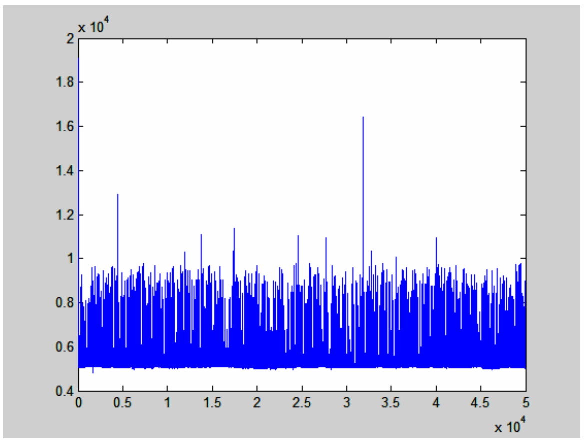


Geode

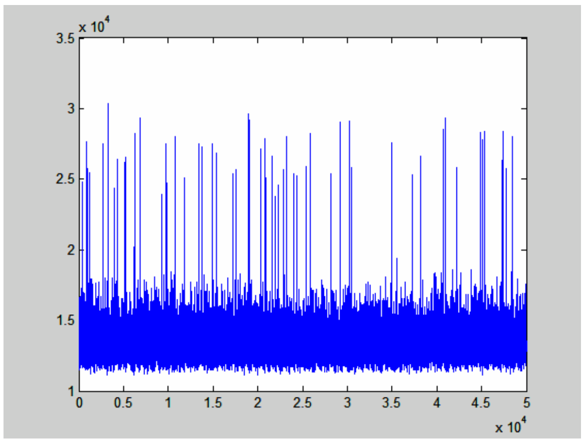
AMD1000

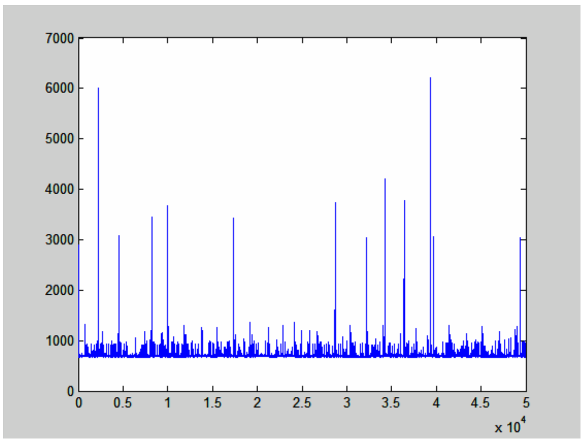
Write



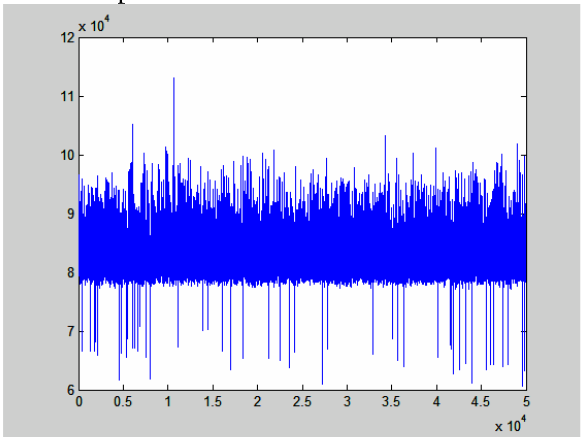


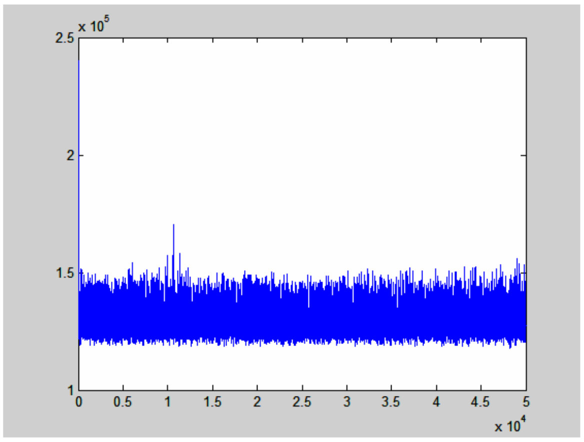
Read





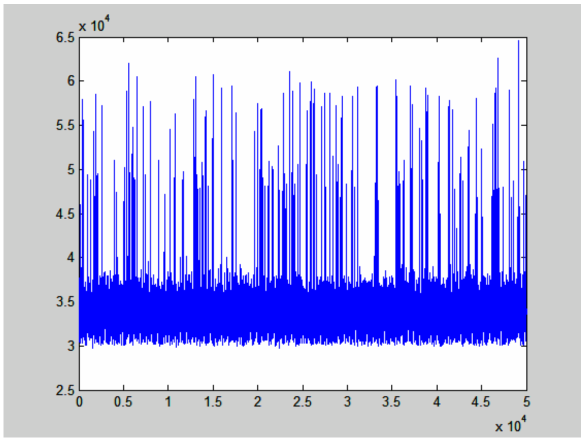
Roundtrip



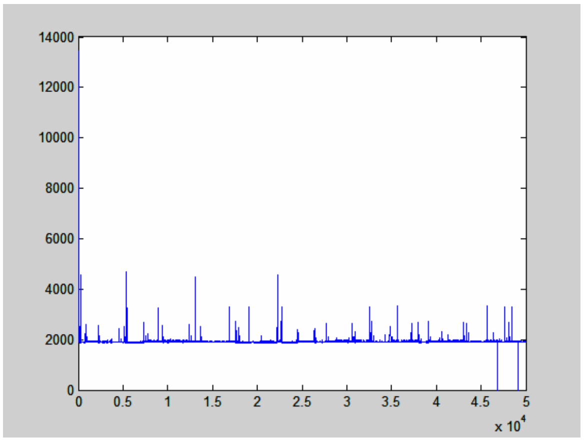


Geode

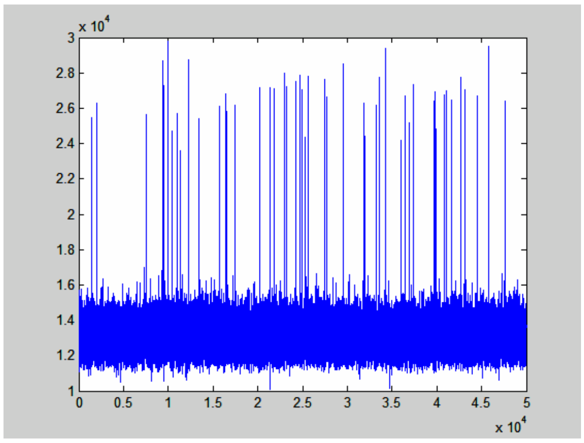
Write

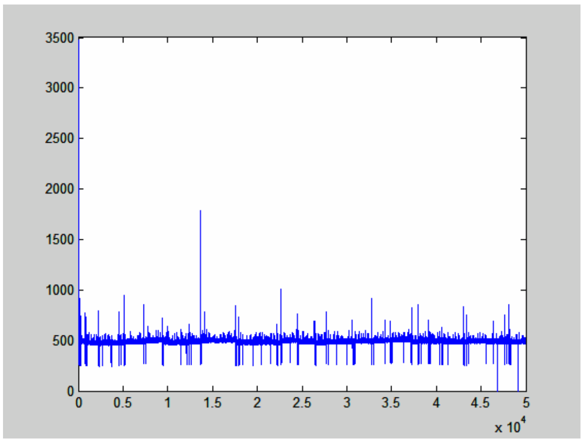


laptop

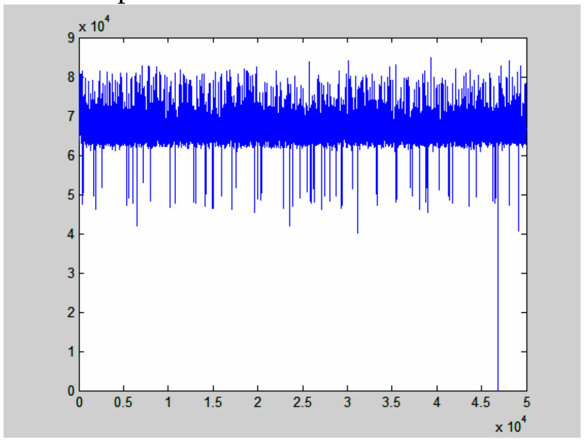


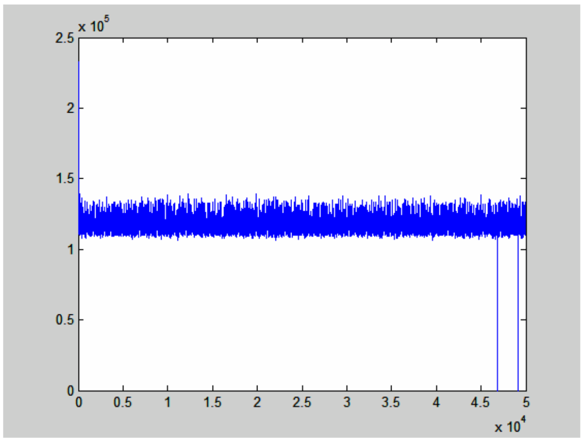
Read





Roundtrip

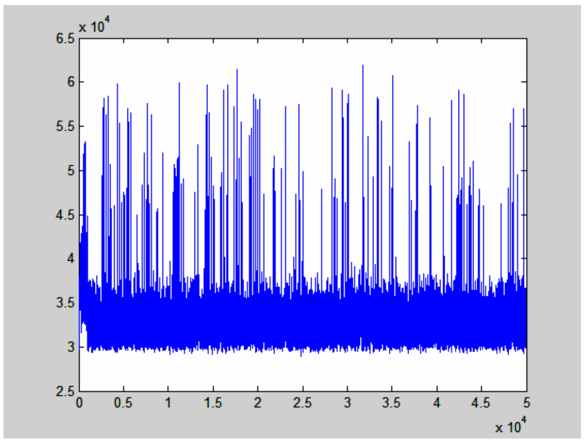


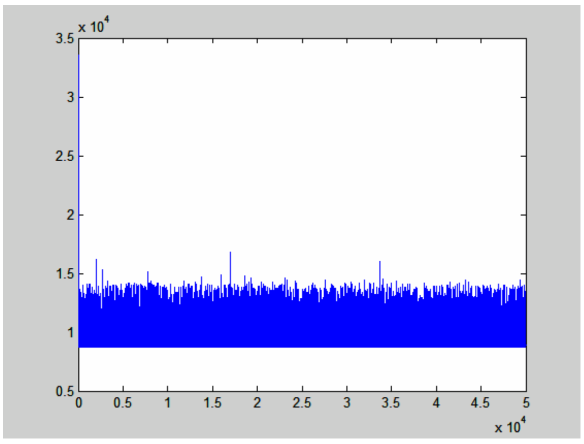


Geode

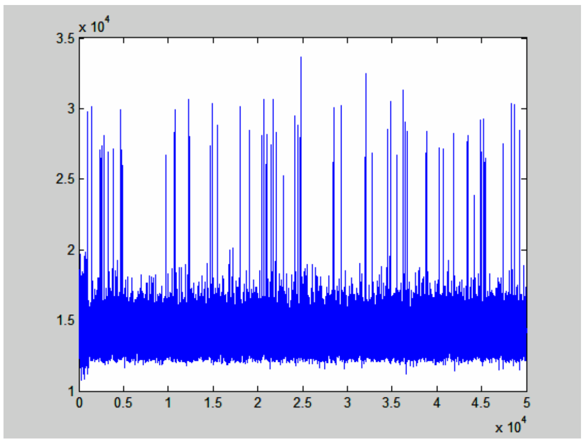
PC104

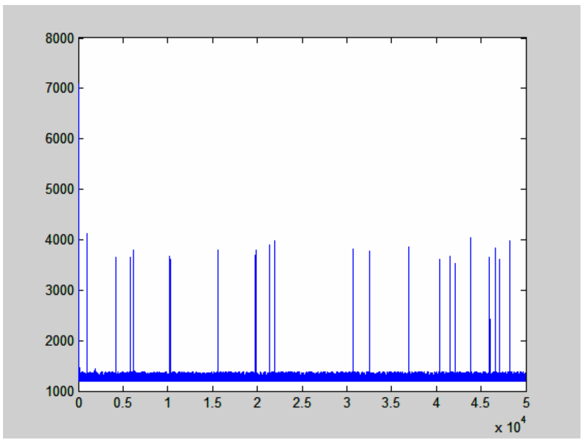
Write



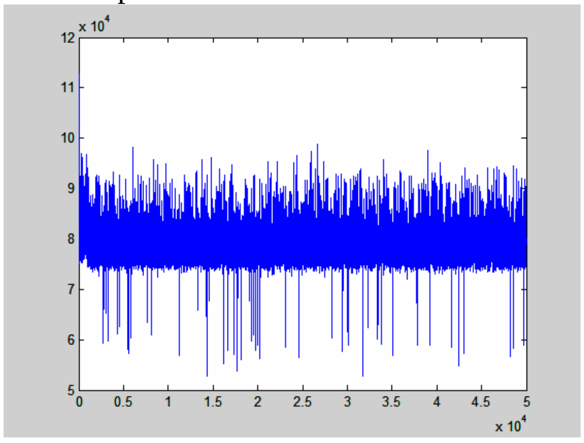


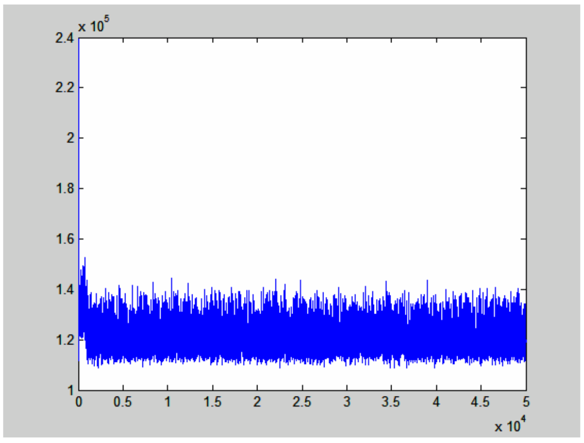
Read





Roundtrip

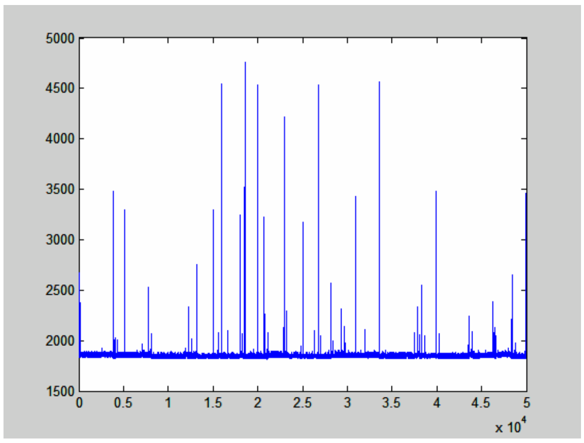


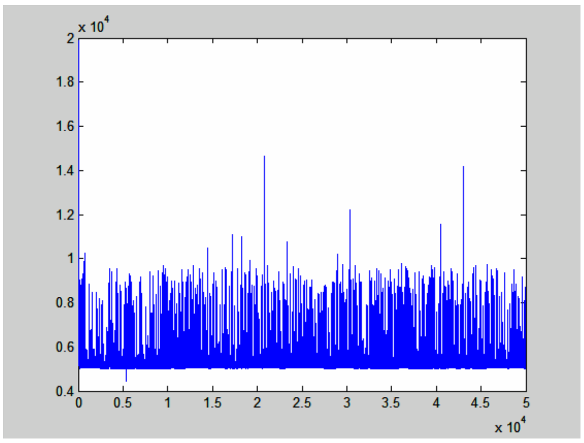


Laptop

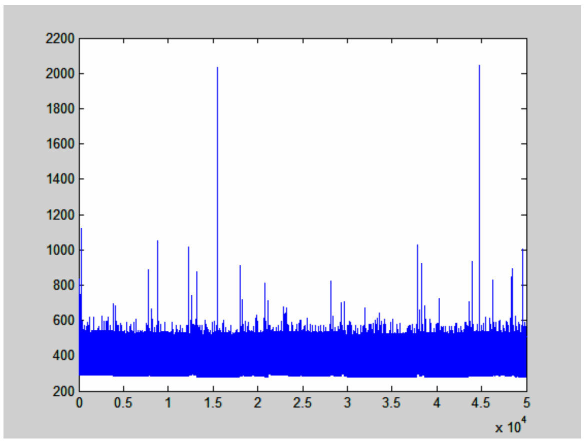
Write

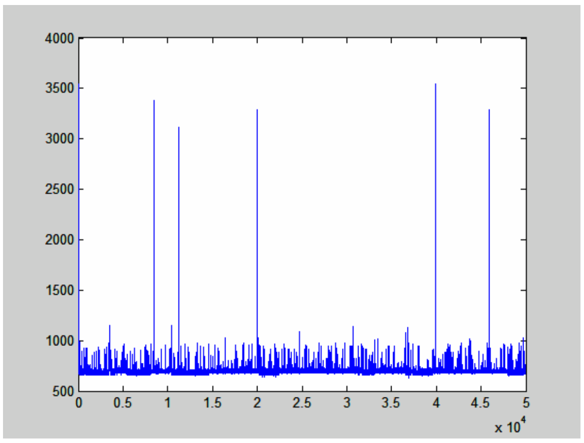
AMD1000



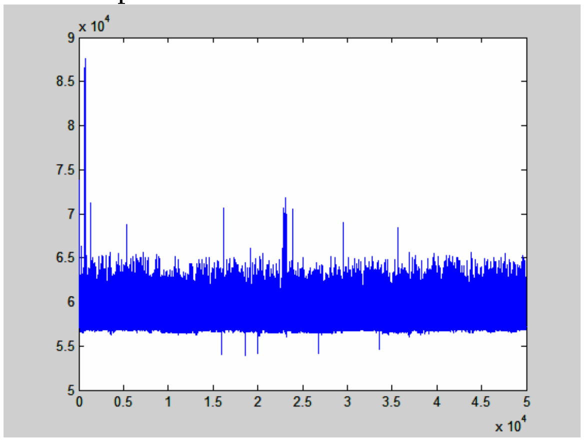


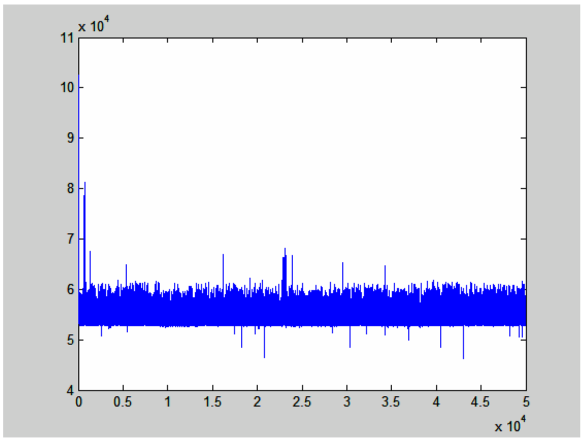
Read





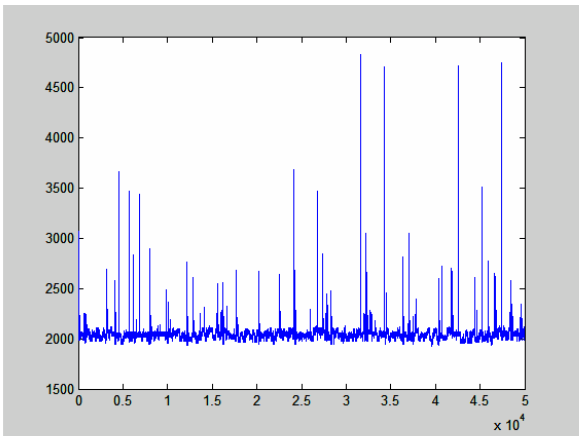
Roundtrip



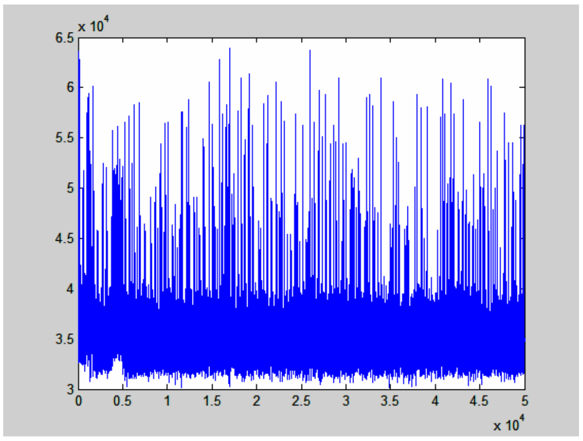


Laptop

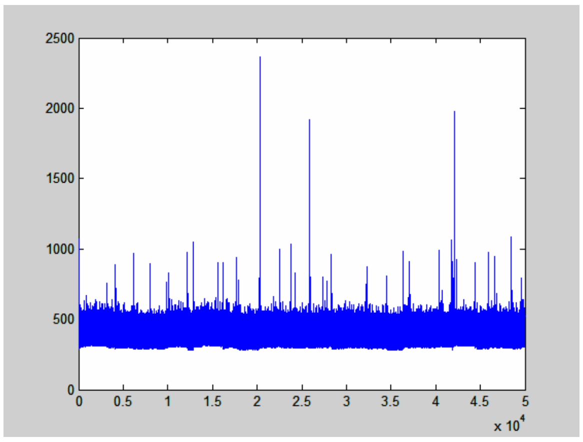
Write

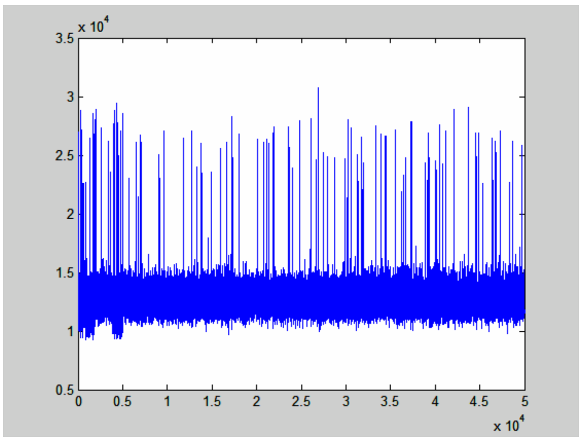


geode

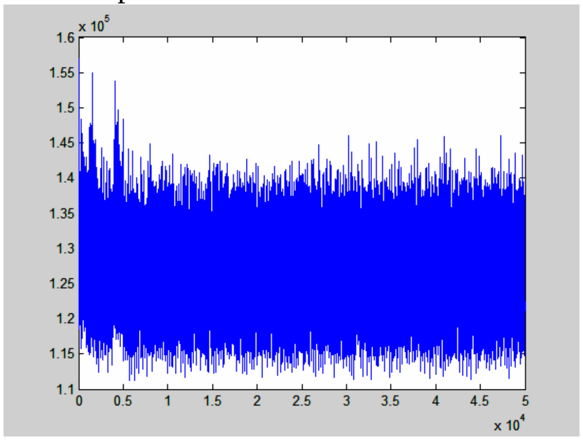


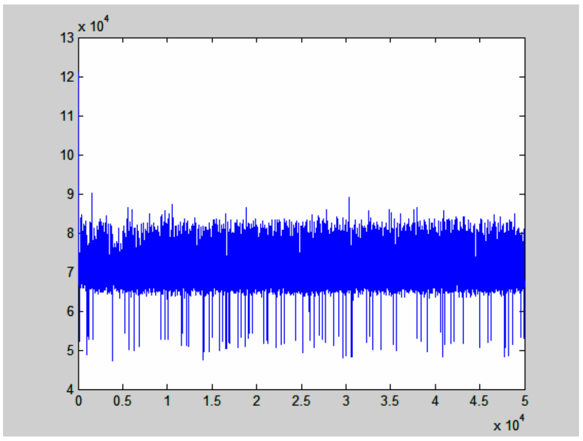
Read





Roundtrip

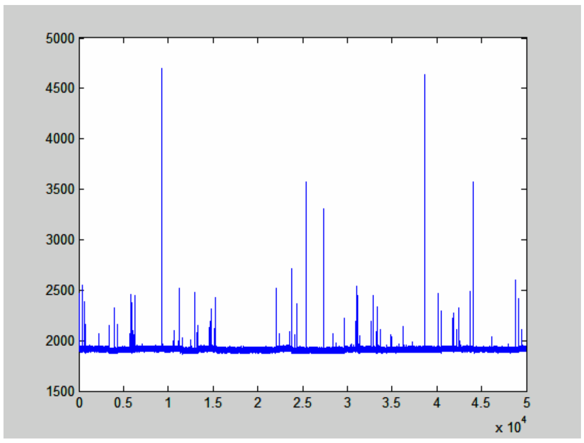


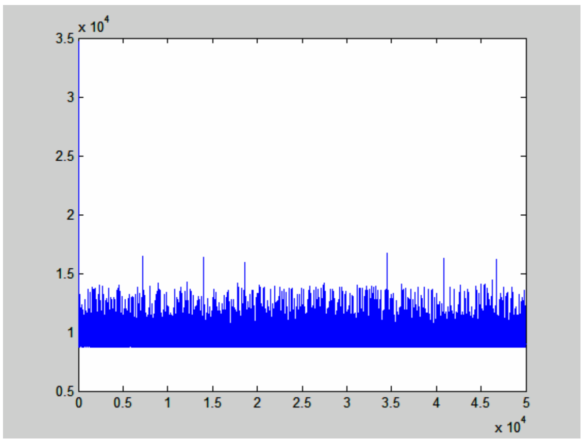


Laptop

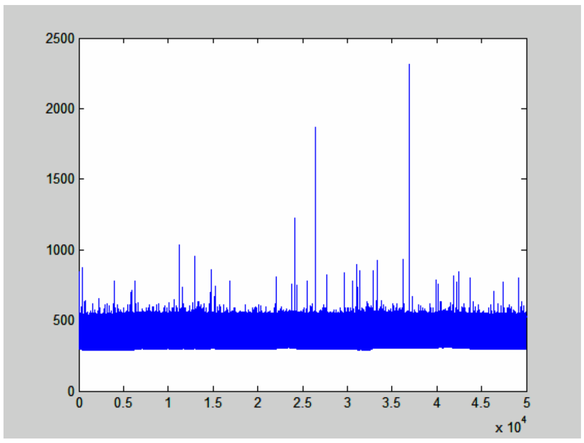
PC104

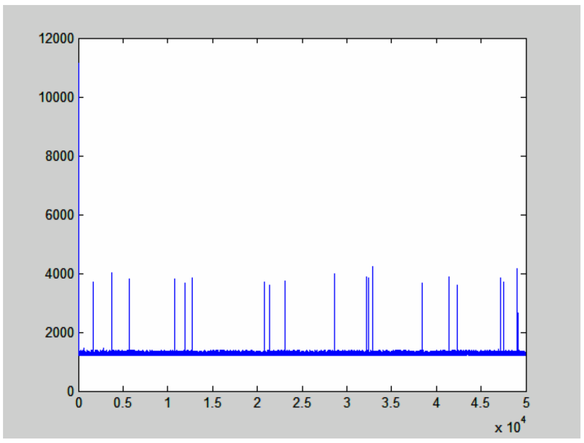
Write



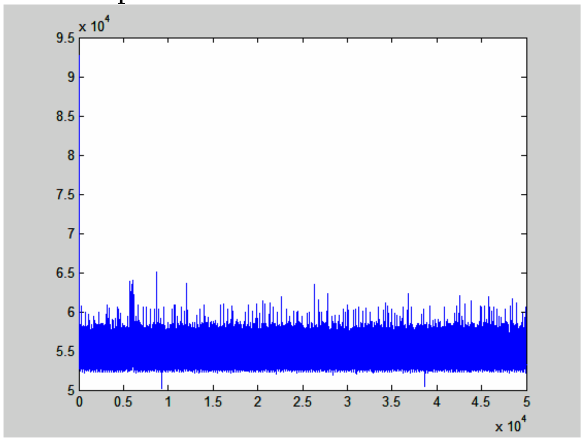


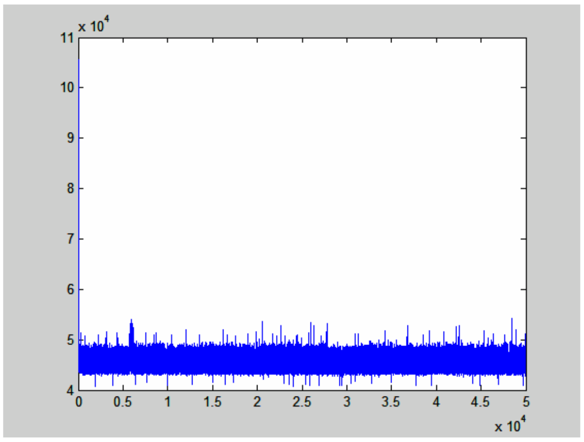
Read





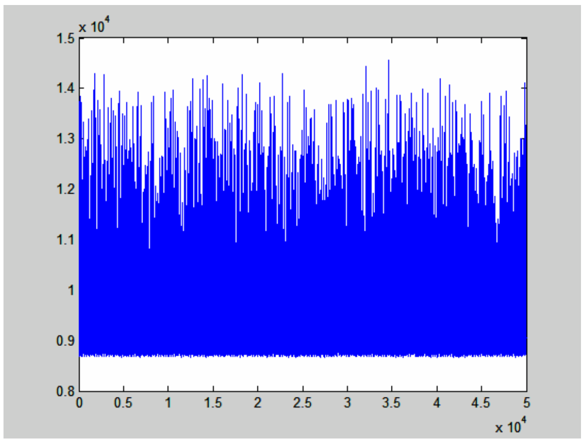
Roundtrip

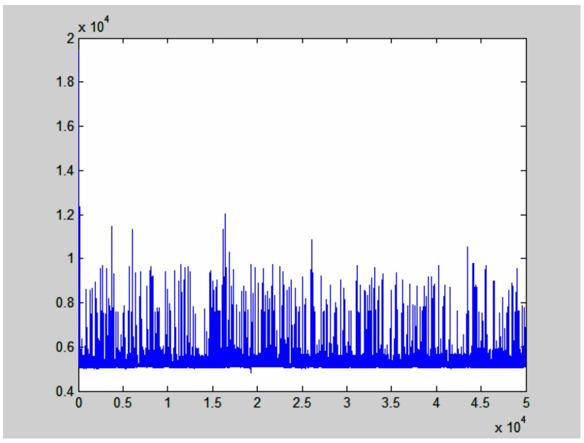




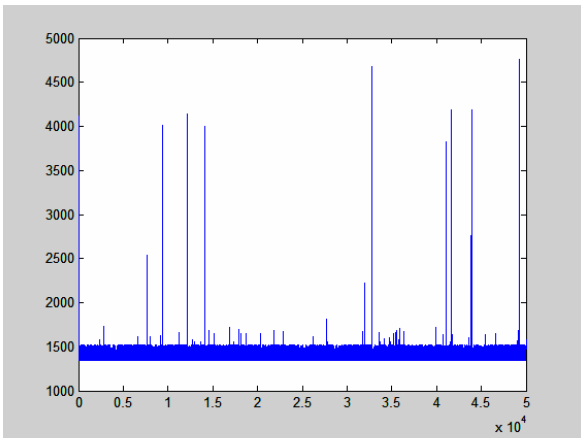
Pc104 Amd1000

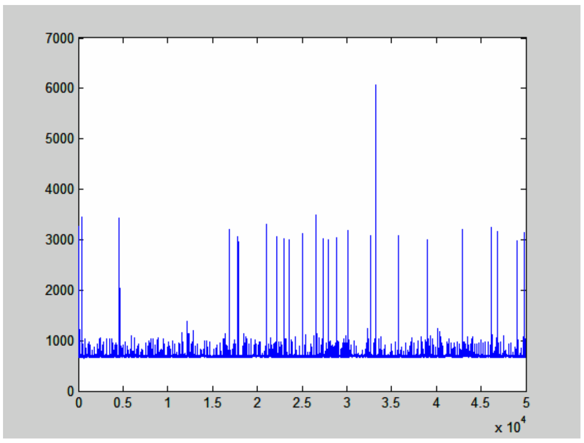
Write



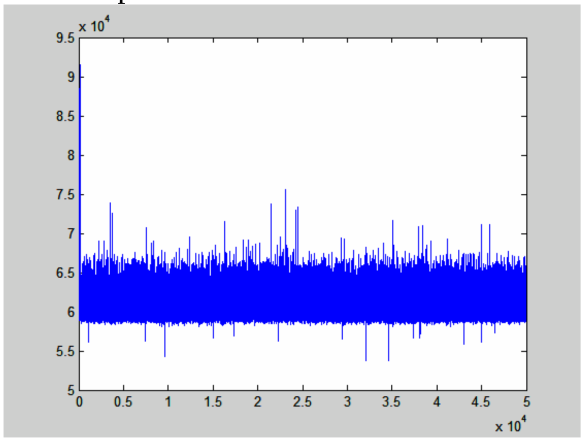


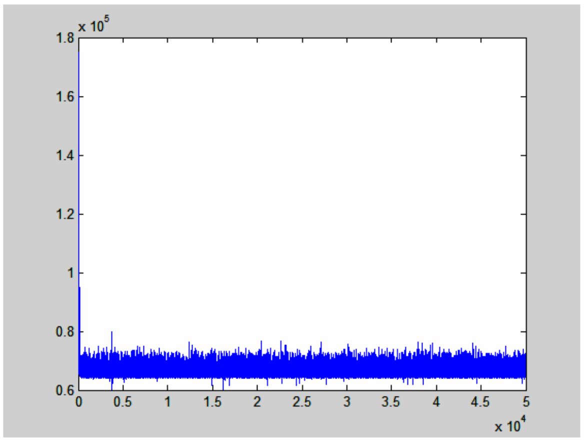
Read





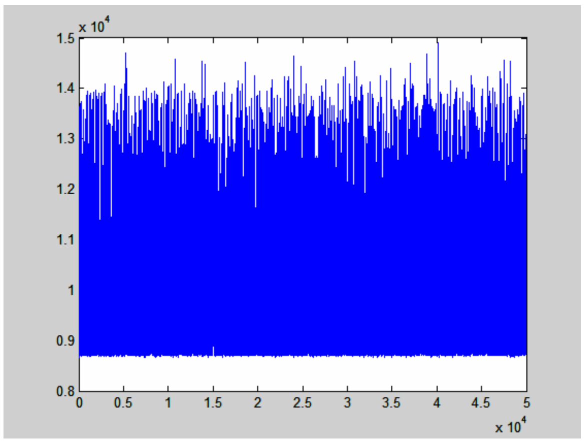
Roundtrip

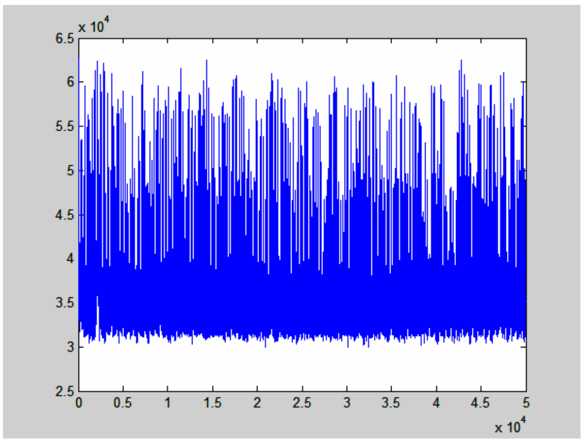




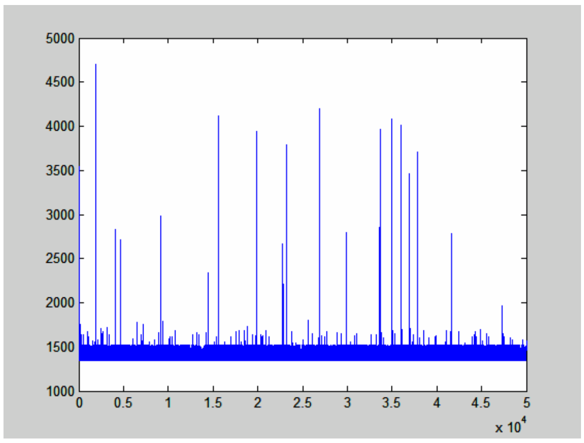
Pc104 geode

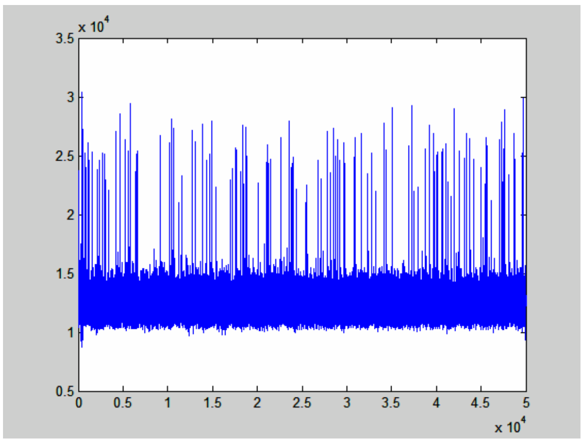
Write



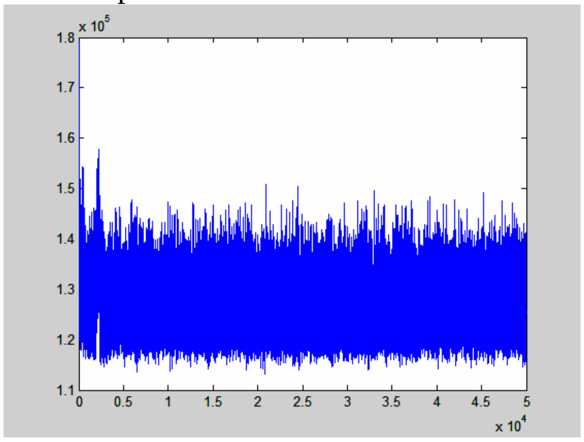


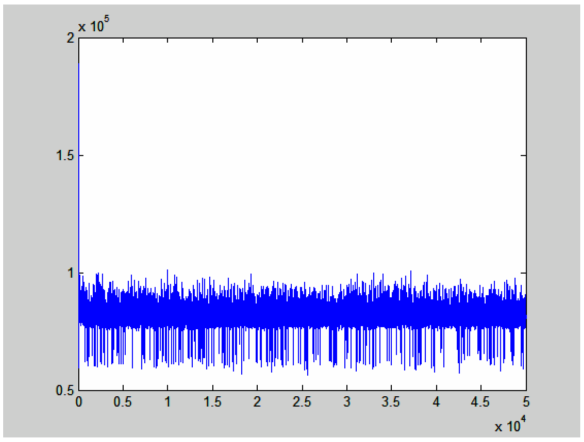
Read





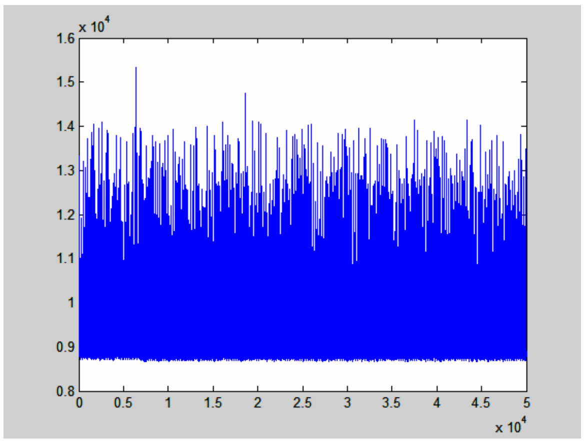
Roundtrip

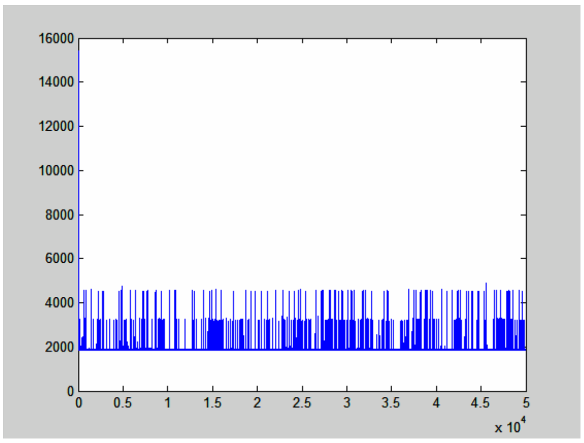




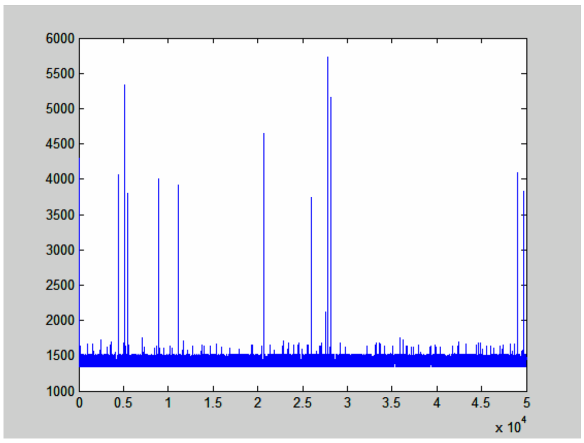
Pc104 laptop

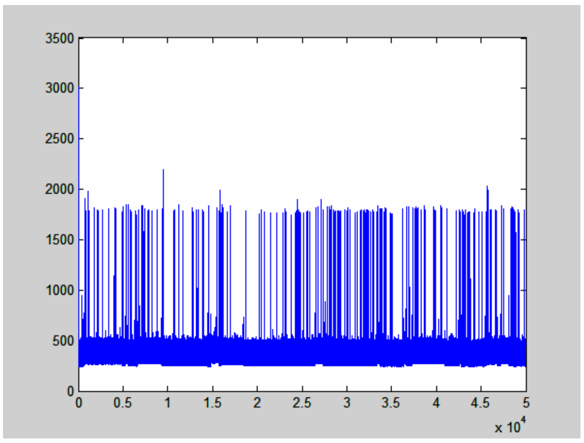
Write



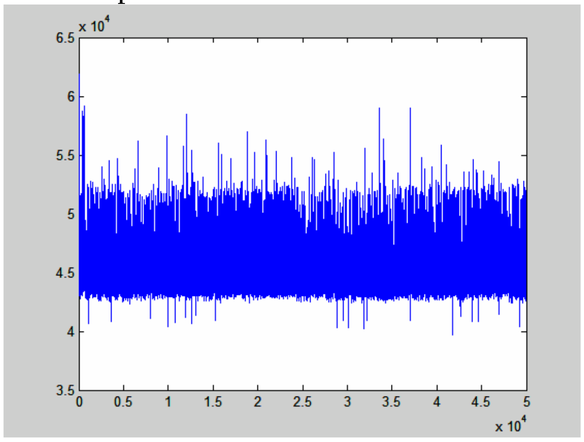


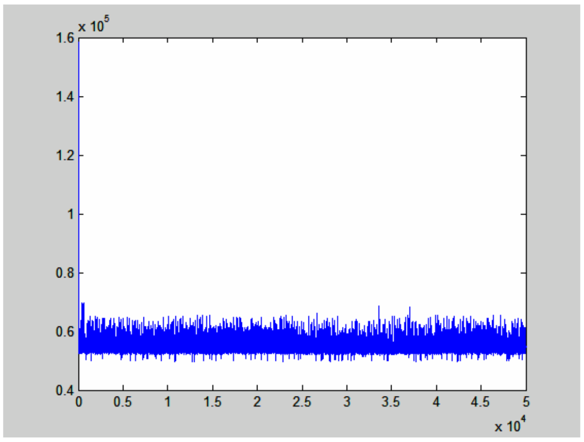
Read





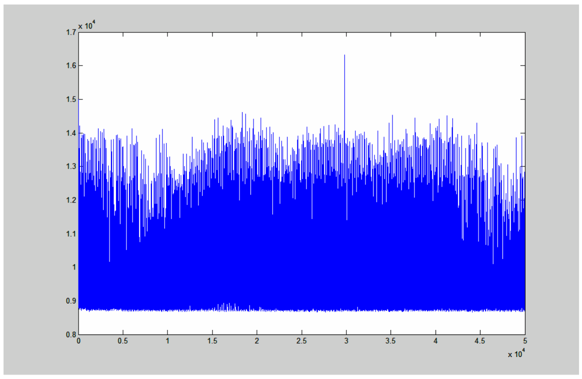
Roundtrip



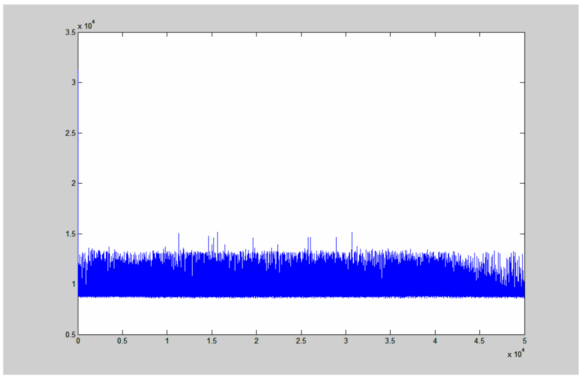


Pc104

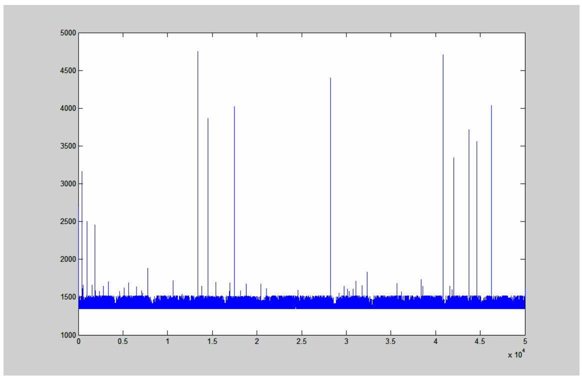
Write

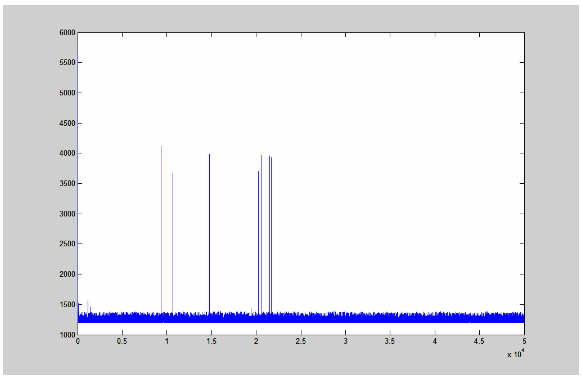


Pc104

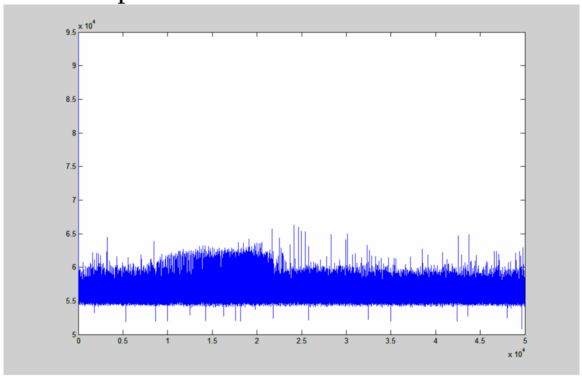


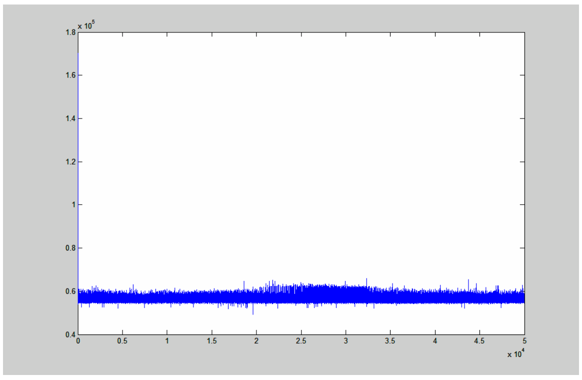
Read





Roundtrip

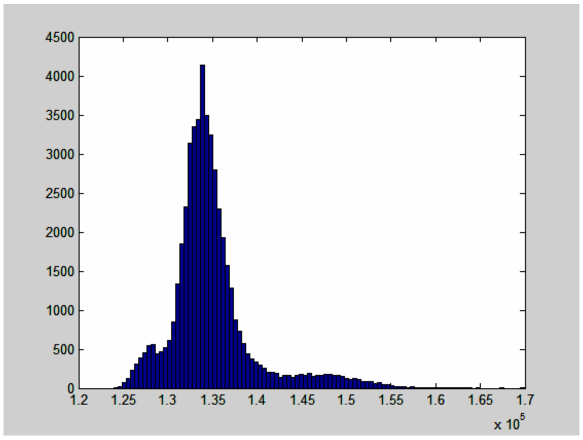




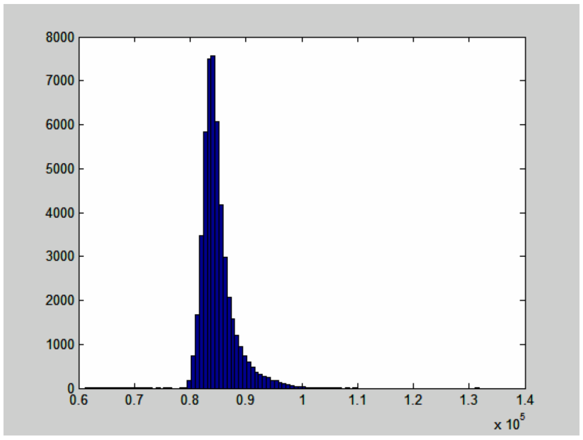
Appendix IV

Distributions of the roundtrip time as they have been measured by the device that is stated above. All graphs are in server/client model, on the left the time measured by the server of the client on the right visa versa. What can be noticed from all graphs is that when a Geode is involved the distributions get wider.

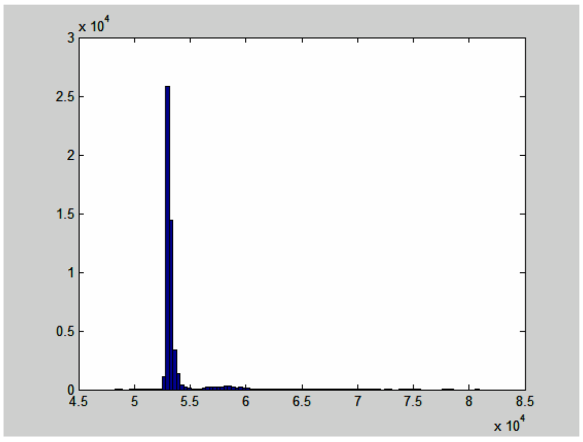
Server:Amd100



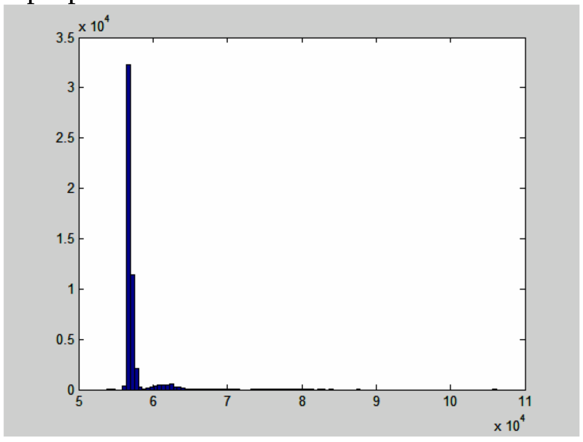
Client:Geode



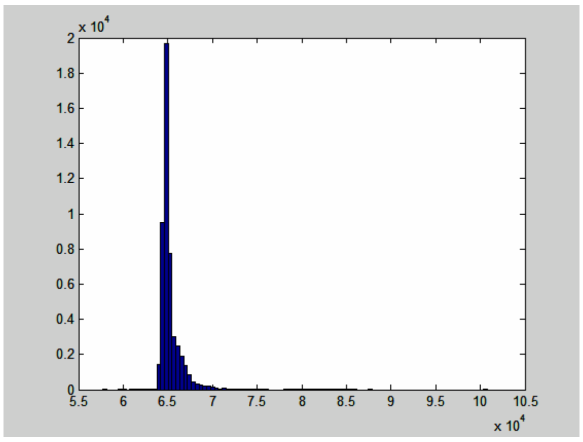
amd1000



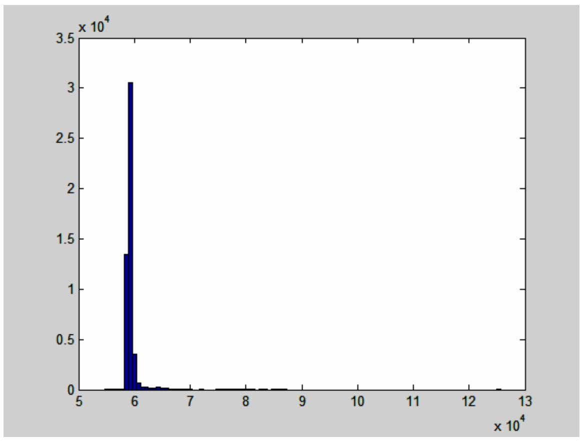
laptop



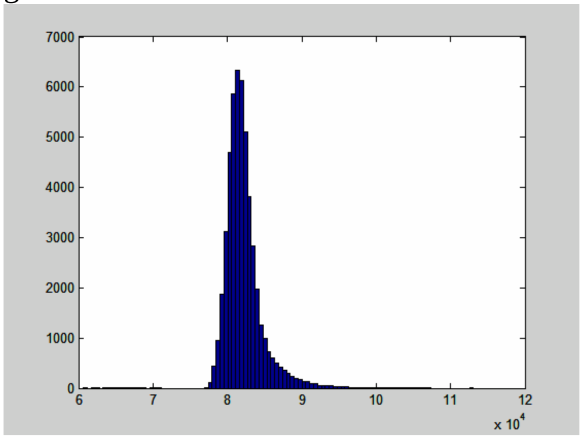
amd1000



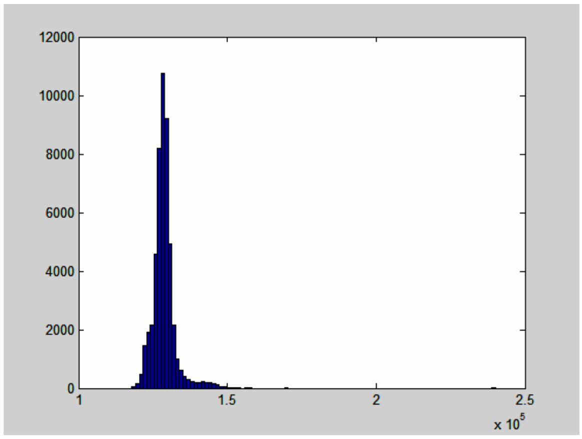
PC104



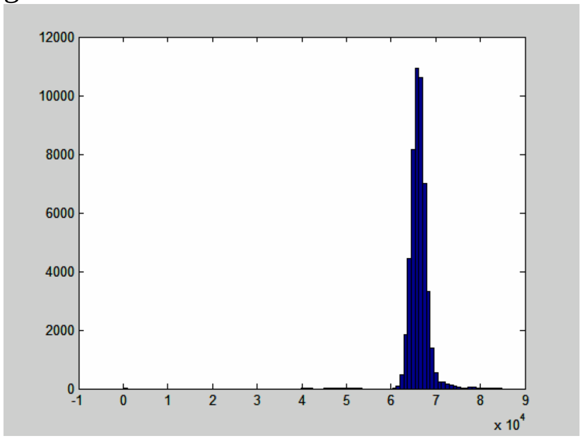
geode



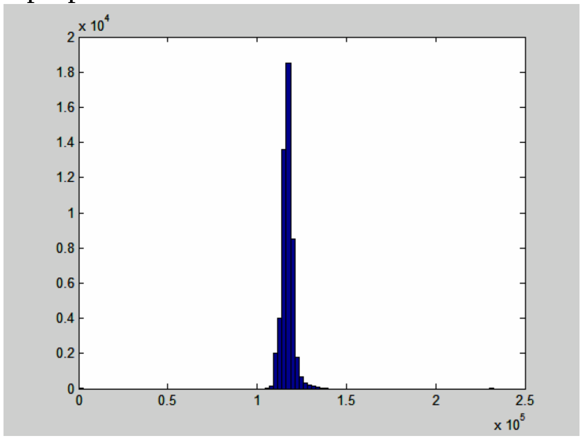
amd



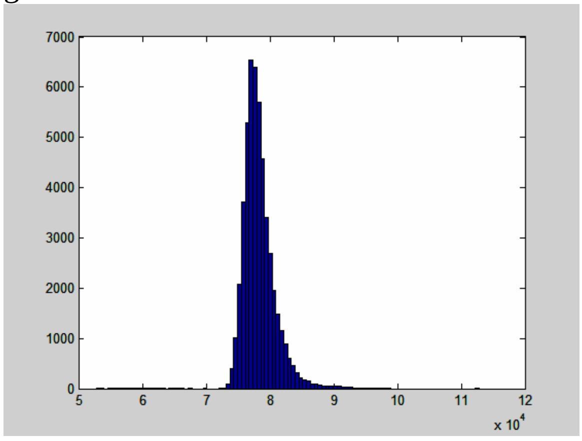
geode



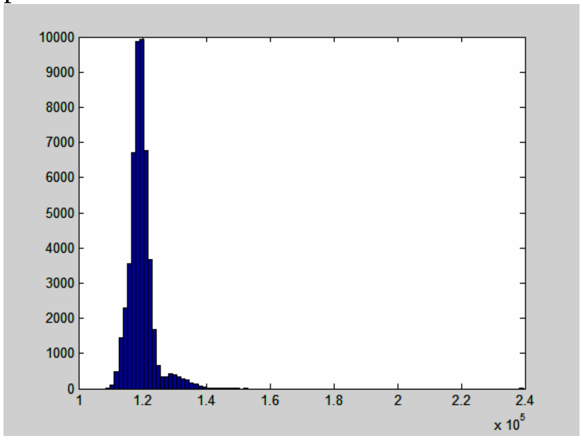
laptop



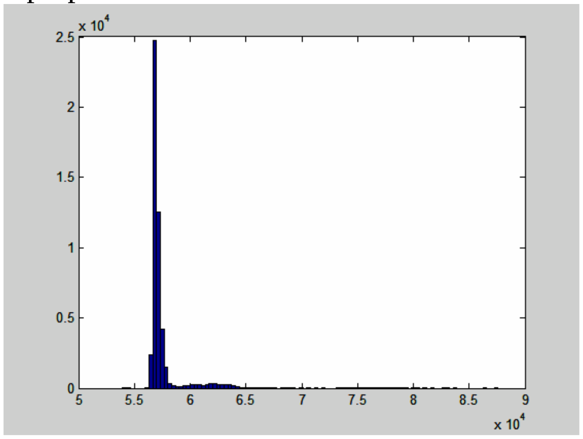
geode



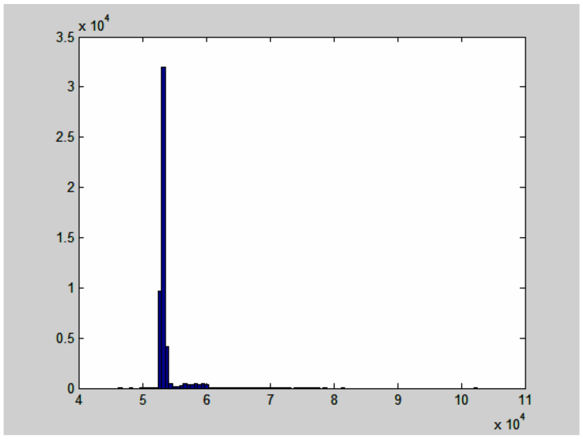
pc104



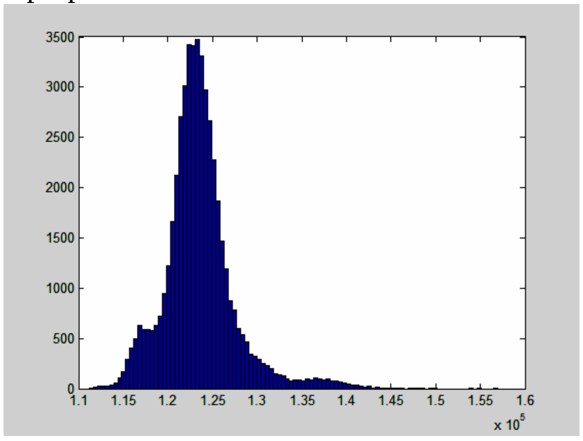
laptop



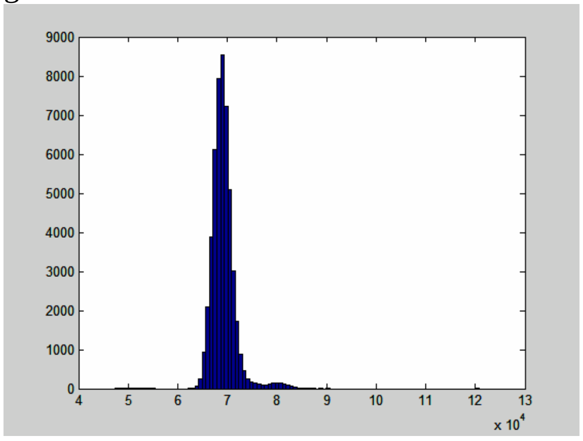
amd1000



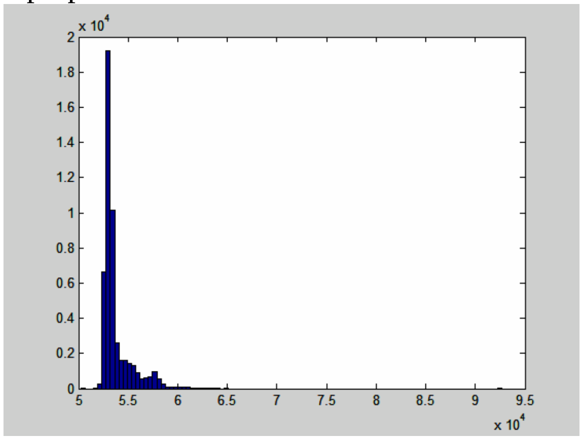
laptop



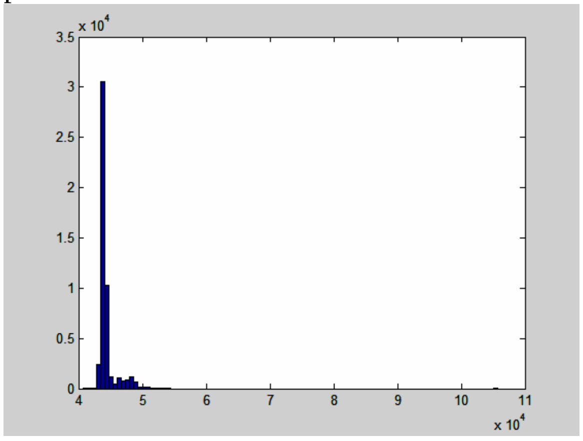
geode



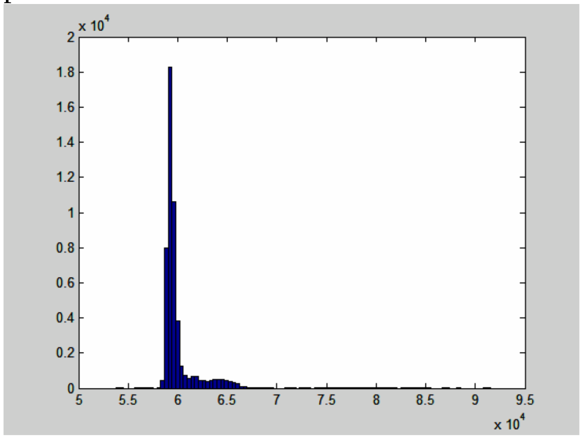
laptop



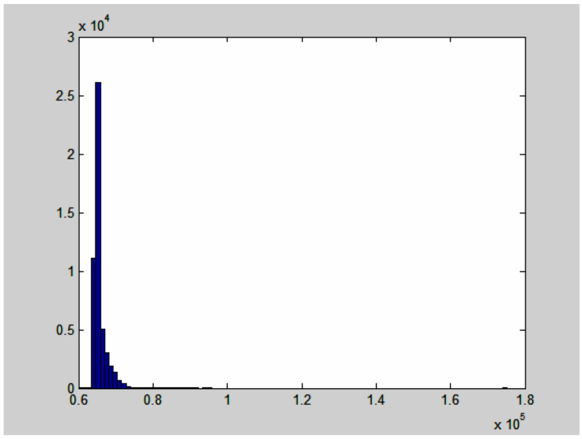
pc104



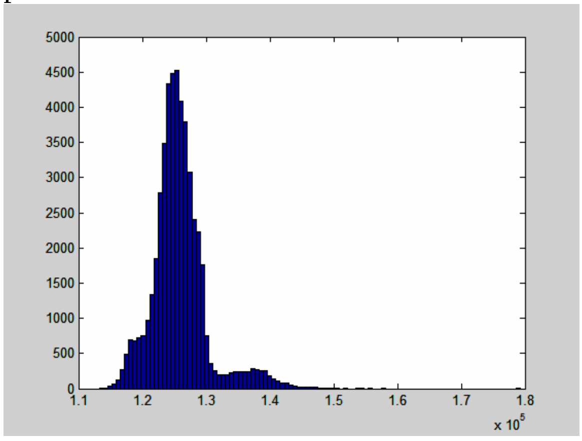
pc104



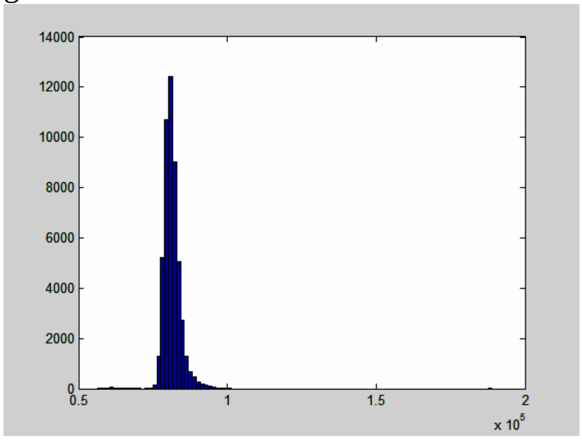
amd



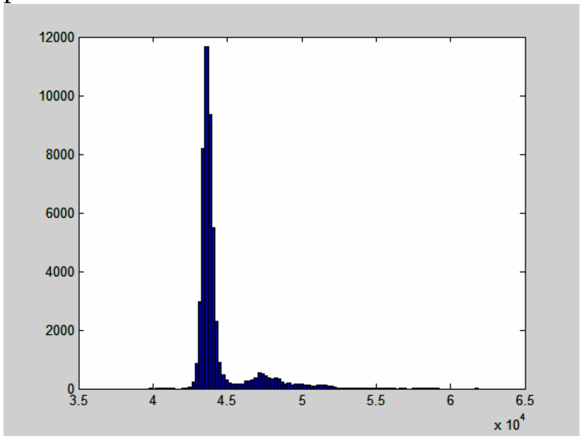
pc104



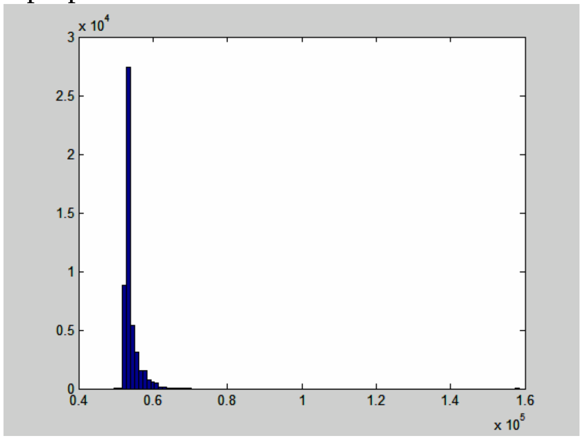
geode



pc104



laptop



References

Locamation (2004), *Locamation homepage, http://www.locamation.com*

RTAI (2004), *RTAI homepage, http://www.rtai.org*

Engelen (2002), *Stageverslag T.H. van Engelen, Control Labratory, Univeristy of Twente ???CE200?*

Hischer (2004), *Hilscher homepage, http://www.hilscher.com*

Stork (2004), *Undocumented theoretical tests, information from Johan Visser (johan.visser@stork.nl)*

Locbus (1999, 2001), *manual LC310_01.P65 and manual PI300 Series, Locamation manuals*

RTnet (2004), *RTnet homepage, http://www.rtnet.org*