ON THE ETHERNET USE FOR REAL-TIME PUBLISH-SUBSCRIBE BASED APPLICATIONS

Ondrej Dolejs
Czech Technical University in Prague
FEE, DCE, Center for Applied Cybernetics
Karlovo nam. 13, 121 35 Prague
Czech Republic
xdolejs@fel.cvut.cz

Petr Smolik, Zdenek Hanzalek
Czech Technical University in Prague
FEE, DCE, Center for Applied Cybernetics
Karlovo nam. 13, 121 35 Prague
Czech Republic
xsmolik ,hanzalek @fel.cvut.cz

Abstract

Ethernet is very attractive for the automation area due to its availability and low implementation cost. Because of its media access control (CSMA/CD), Ethernet is not deterministic in general and its behaviour under transient overload is not sufficient for any real-time application. On the other hand, if the applications have predictable and bounded number of requests, behaviour of Ethernet is "nearly" real-time (very low probability of delayed data delivery).

This article tests ORTE (Open Real-Time Ethernet), an open-source implementation of RTPS middleware (Real-Time Publish-Subscribe), built upon UDP/IP and tested on Ethernet. This middleware can be used in real-time control applications, which typically have limited and relatively small input load compared to the high bandwidth. To derive the influence of the operating system, we combined the application response time measurement with the simulation.

1. Introduction

Ethernet was designed for computer networks in 70's without any real-time requirements and it was normalised later as IEEE 802.3 standard. Today, Ethernet is widely spread in office computer communication, which implies high availability and low implementation cost of Ethernet based solution. Therefore Ethernet is very attractive for the automation area due to high performance and wide diagnostics possibilities (using several protocols simultaneously running on the same medium).

Distributed real-time applications have two main requirements for data delivery: time determinism and reliability. Ethernet in general is not deterministic due its media access control (CSMA/CD) and therefore its behaviour under transient overload is not sufficient for any real-time application. On the other hand, if the applications have predictable and bounded number of requests, the behaviour of Ethernet is "nearly" real-time — the probability of the delayed data delivery is very low [6] due to the reasonably low number of accesses compared to the high-bandwidth performance.

Widely used and popular protocol TCP/IP is reliable, but it cannot provide time determinism. The number of retried packets is not predictable and causes the time non-determinism. The UDP/IP protocol is better suited for distributed real-time applications, since it is deterministic (in the sense of packet retransmission). The UDP/IP needs to be combined with an upper layer, which provides packet retransmission suited for control applications due to the UDP/IP's insufficient reliability.

Several communication protocols using Ethernet for automation came up in the last years. Some of them are built on the top of TCP (Modbus TCP, ProfiNet v.1.2), some on the top of UDP (NDDS - Network Data Delivery Service, Modbus UDP) and some are built directly on Ethernet (PowerLink, ProfiNet v.2). NDDS middleware [4] is commercial implementation of RTPS (Real-Time Publish-Subscribe) protocol, see [3]. ORTE (Open Real-Time Ethernet), our open-source implementation of the RTPS specification, is an alternative to NDDS built on the top of the UDP/IP protocol and tested on Ethernet, freely available at (http://sourceforge.net/projects/ocera), see [7].

PowerLink, developed by Bernecker&Rainer company, is a proprietary real-time protocol built directly on the top of the Ethernet link layer. This protocol use the Time-Division scheme for data delivery, where every node has a time-slot to send its data to avoid the collisions on Ethernet.

The problem of the data delivery probability estimation for NDDS can be partially solved by analytical methods using probability calculation [6] or by experimental results as is in [5] and [10].

This article tests ORTE (Open Real-Time Ethernet), an open-source implementation of RTPS middleware (Real-Time Publish-Subscribe), built upon UDP/IP and tested on Ethernet. To derive the influence of the operating system, we combined the application response time measurement with the simulation.

This paper is organized as follows. Section 2 deals with the basics of the Ethernet behaviour and it shows application response time, which is a crucial parameter for real-time applications. A simulation approach is used in Section 2, since it offers more flexibility, especially for large networks. Section 3 explains the Real-Time Publish-Subscribe (RTPS) protocol, and shows a simple application of ORTE, our open-source implementation of RTPS. Section 4 focuses on the experimental results measured on a given configuration.

2. Preliminaries on Ethernet

The time analysis of the real-time behaviour of Ethernet and IP was done in [6]. The CSMA/CD (Carrier Sense Multiple Access/Collision Detection) is a well known media access method (MAC) used by Ethernet (IEEE 802.3). A collision can arise when at least two waiting nodes want to send data at the same time. These two transmission attempts are usually accumulated during the previous channel activity. The probability of N collisions in this configuration is given as follow:

$$P_{err} = \prod_{1}^{N} \frac{1}{2^n + 1} = \frac{1}{3} \times \frac{1}{5} \times ..$$

At the first attempt, two nodes choose randomly a number from the range {0,1,2}. Therefore the probability that they choose the same number is 1/3. If a collision occurred, the range is doubled, and the nodes can choose from 5 numbers {0,1,2,3,4}. The range is doubled several-times (at most 10 times). The range is interval of integers from 0 to 2n. This mechanism is known as exponential back-off. The back-off delay is equal to the chosen random number multiplied by the slot time. In Ethernet the slot time is 51,2 microseconds for 10Mbit/s and 5,12 microseconds for 100Mbit/s. With this knowledge it is possible to count the probability and the time for different numbers of slots. From the real-time point of view a critical situation occurs if the communication time exceeds some given deadline (real-time error). The deadline can be expressed as a function of N, since the communication time is composed of the time spent for the media access and a constant packet propagation delay. Subsequently P_{err}, the probability of the real-time error occurrence, is approximately equal to the probability of N collisions, where N is the minimum number of collisions needed to exceed the deadline. There is another drawback of Ethernet making it unsuitable for hard real-time applications - the packet dropping after 16 unsuccessful attempts to access the bus. On the other hand, Ethernet can be used for large group of soft real-time applications, where e.g. the loss of packet, containing a periodically sampled temperature is not crucial for the system behaviour.

In the rest of this article, we have used simulation in OPNET and experiments as an alternative to analytical analysis of Ethernet behaviour [2]. The OPNET simulation software is based on a series of hierarchically related editors, e.g. Network, Node and Process editors that directly model the structure of actual networks. This simulation software enables simple changes in the network configuration, e.g. the number of nodes, the bandwidth, and the packet size.

Figure 1 shows a configuration of 10 sending nodes and one receiving node connected via Ethernet (100 Mbit/s) to a hub. This configuration was chosen only to achieve the necessary input load. The same application is running in each sending node and it generates the required input load [packet/s], which is defined [2] as a sum of all packets send by application to the lower layers. The packet length is equal to 128 bytes, which is a sufficient length in communication among sensors, controllers and actuators.

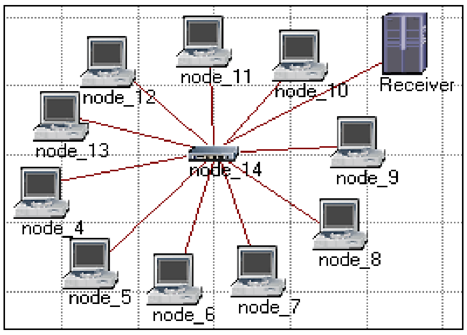

Figure 1 Simulation configuration

Figure 2 illustrates a histogram for simulations of six different input loads for the configuration shown in Figure 1. Thus it shows a distribution function of a communication time (consisting of the media access time and transmission time). The distribution function is important for real-time control applications, since one can simply derive the count of packets exceeding a given deadline.

For lower values of the input load (up to 30000 packets/s, i.e., each of 10 sending nodes transmits 3000 packets/s) the communication time of all packets is in a very tight interval of around 100 µs. For the input load of 40000 packets/s, the communication time of considerable part of packets is still around 100 µs, but there are also packets with communication time of around 2ms. For the input load of 50000 packets/s, the communication time is distributed in a rather wide

interval from 1ms to 20ms, and for 66666 packets/s the communication time ranges from 10ms to 1s.
Roughly speaking, it is clear that 100Mbit/s Ethernet can be used up to the input load of 30000 packets/s for a large set of real-time control applications. Further test are in [11].

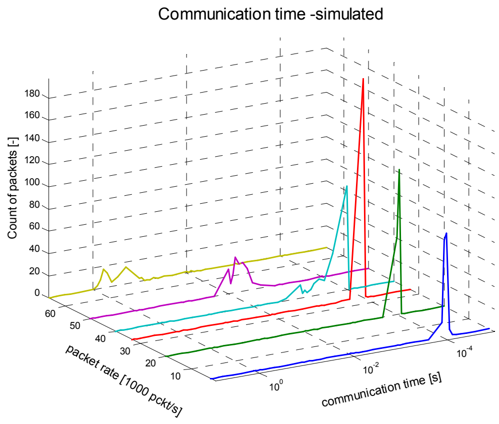

Figure 2 The communication time histogram in relation to Input-load

3. Real-Time Publish-Subscribe protocol

Real-time applications require more functionality than is provided by the traditional publish-subscribe semantics. Real-Time Publish-Subscribe protocol (RTPS) [3] adds publication and subscription timing parameters and properties so that the application developer can control different types of data flows and therefore the application's performance and reliability goals can be achieved.

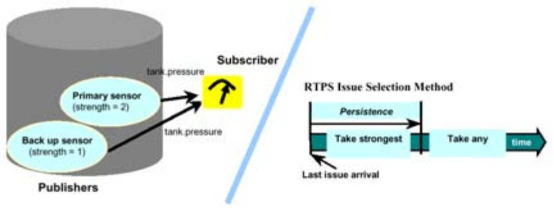

Figure 3 Publisher parameters

The timing parameters, shown in Figure 3 and Figure 4, have the following meaning. Publication parameters (see Figure 3): *topic* (name of publication) *and type* (message type) identify a specific publication; *strength* is the relative weight (priority) of the publication compared to the publications of the same topic and type; *persistence* specifies how long a publication is valid. When the persistence elapses, the subscriber takes the first received publication.
Subscription parameters (see Figure 4) *topic & type* identify a specific publication; no new publication is accepted during *minimum separation* time; *deadline* specifies how long a new publication is expected. When the deadline for the data delivery is passed without receiving any publication, then the application is informed (timeout notification). The user can set all parameters to fulfil the application requirements.

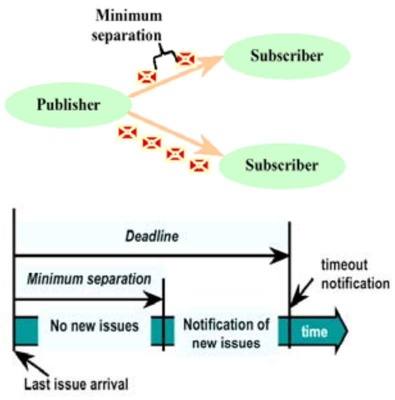

Figure 4 Subscriber parameters

3.1. ORTE implementation

The open source implementation of the RTPS protocol has been done at the Czech Technical University in Prague as one result of the OCERA project [9]. Although the object concept of RTPS would be ideal for an object oriented programming language such as C++, the final ORTE implementation is done in C language, since it allows simple porting of ORTE to different operating systems, mainly those with the real-time behaviour. Figure 5 and Figure 6 show how simply ORTE can be used.
In order to exchange user data, the application must create publications of its variables. An application willing to receive publications of published data must create a subscription. Properties of the publication and the subscription contain specifications of *Topic* and *Type*, which specify an application variable within whole network. It is allowed to have more publications of the same *Topic* and *Type (see* Figure 5*)*.

```
ORTEPublication *p;
NtpTime persistence, delay;
ORTEInit();
d = ORTEDomainAppCreate(ORTE_DEFAUL_DOMAIN,
NULL, NULL, ORTE_FALSE);
if (!d)
ORTETypeRegisterAdd(d, "HelloMsg",NULL,
                                    NULL,64);
NTPTIME_BUILD(persistence, 3);//is valid for 3s
NTPTIME_DELAY(delay, 1);
p = ORTEPublicationCreate(
  d,    // pointer to application object
  "Example HelloMsg",  // name of topic
  "HelloMsg",          // data type description
  &instance2Send,      // output buffer
  &persistence,// persistence of publication
  1,   // strength of publication
  sendCallBack,//pointer to callback function
  NULL,//user parameters for callback
    &delay);// period for timer, callback
```

Figure 5 The skeleton of the ORTE publisher

The subscribing application needs to create a subscription with publication's *Topic* and *Type*. A callback function is called whenever a new publication from the publisher is received.

```
ORTESubscription *s;
NtpTime deadline, minimumSeparation;
ORTEInit();
d   =   ORTEDomainAppCreate(ORTE_DEFAUL_DOMAIN,
NULL, NULL, ORTE_FALSE);
if (!d)
ORTETypeRegisterAdd(d, "HelloMsg", NULL, NULL,
64);
NTPTIME_BUILD(deadline, 20);
NTPTIME_DELAY(minimumSeparation, 0);
p = ORTESubscriptionCreate(
  d,   // created subscribtion
  IMMEDIATE,   // mode of subscription
  BEST_EFFORTS,// type of subcsription
  "Example HelloMsg",// name of topic
  "HelloMsg",// name of data type
  &instance2Recv,// pointer to output buffer
  &deadline,   // deadline
  &minimumSeparation,// minimum separation
  recvCallBack, // callback function
  NULL); user parameters
```

Figure 6 The skeleton of the ORTE subscriber

The initial implementation has been developed on Linux kernel 2.4, but it is able to run on both 2.2 and 2.6 versions as well. ORTE is designed as a library that allows simple linking with the user application.The current ORTE version was tested under Linux and Windows 2000/XP. In future, we intend to test ORTE on RTAI based on RTNET (special UDP implementation for RTAI) and RTLinux. The ORTE source code can be downloaded from [8] and documentation is available at [9].

4. Measurements of the application response time

Important requirements in real-time applications are the application response time and throughput. The application response time is the time measured from the moment, when the application calls the middleware to send a publication through the network layers, to the time, when the subscribed applications get this publication. The application response time can be affected by the network load, the network bandwidth, the implemented network stack, the processor speed, and in most cases by the operating system.

This section shows experimental results, which have been obtained from the testing of ORTE. Configuration with one switch and four identical nodes (PC's, with Intel Celeron 566MHz, 192MB RAM, 100Mbit Ethernet Card, running Linux Debian 2.6.5 with pre-emptive kernel) was used for these experiments (see Figure 7).

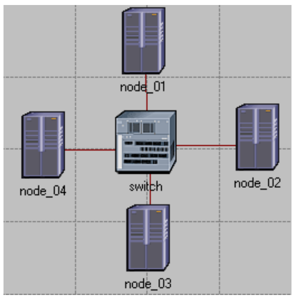

Figure 7 Measurement configuration

The application configuration is given in Figure 8. Publisher P1 in node_1 publishes a publication (containing publication creation time t1), which is received by subscriber S2 in node_2. The publication creation time t1 is extracted and is submitted for publication created by publisher P2 in node_2. This publication is received by subscriber S1 in node_1 at time t2, and then the application response time is calculated, as (t2-t1)/2, and saved for further analysis.

The application response time consists of three parts. The first part is the processing time caused by the operating system overhead, including the scheduling time, the context switch and the propagation through the SW layers such as UDP, IP. The second part is the communication time (the media access time and the transmission time) on Ethernet. Finally the last part is similar to the first one, the processing time on the subscriber site.

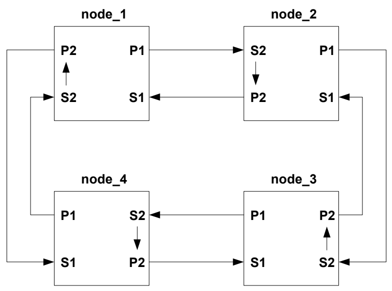

Figure 8 Configuration of publishers and subscribers

A special program to start the application remotely was written to ensure the same starting time at all nodes. The application has the real-time priority using *rtnice*, and stops after sending 5000 packets. The application repetition time (how often is a publication published) and packet size are given to the application as parameters.

4.1. Measured results

The application response time of all send packets is depicted in the Figure 9 and varies from 0.7ms to at most 1.38ms. The mean value is 0.84 ms. This time is composed of three parts as described above.

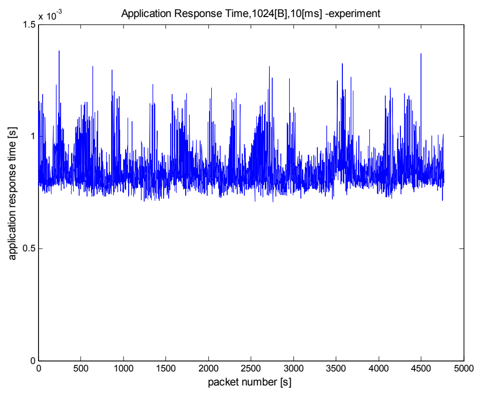

Figure 9 The application response time for packet size 1024B and application repetition time 10ms (configuration in Figure 7)

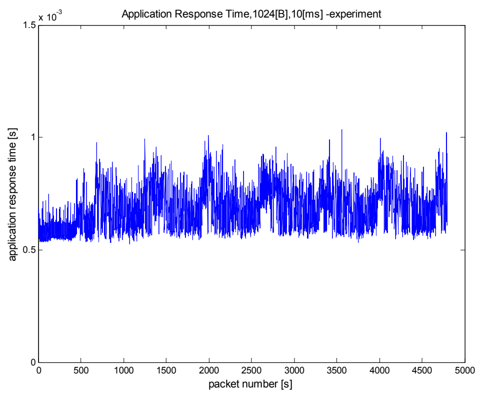

Figure 10 The application response time for packet size 1024B and application repetition time 10ms, configuration 1 nodes

In order to investigate the duration of the separate parts of the application response time we have run another experiment based on one node. The application was the same as described earlier containing two publishers and two subscribers with the same functionality. The packet in this experiment was not transmitted through the Ethernet, but the internal loopback mechanism was used to transmit data from P1 to S2 and from P2 to S1. The results for the application response time in one node are depicted in Figure 10.

The maximum application response time is 1.03ms and the mean 0.67ms. The difference between the mean of the application response time for the configuration with 4 nodes and for the configuration with 1 node is 0.17ms, which is the communication time.

This result was validated by simulation in OPNET Modeler. The HW configuration was made same as depicted in Figure 7 and results for each node are shown in Figure 11.

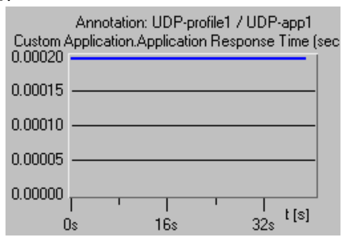

Figure 11 The simulated application response times (configuration Figure 7)

Since the simulation takes only the communication time under consideration (i.e. there is no processing time added to the application response time), therefore the application response time, depicted in Figure 11, is directly the communication time (mean 0.19ms). It affirms the hypothesis given above that the communication time is 0.17ms. A simple calculation shows that the communication takes only approximately 25%, the remaining 75% is the processing time.

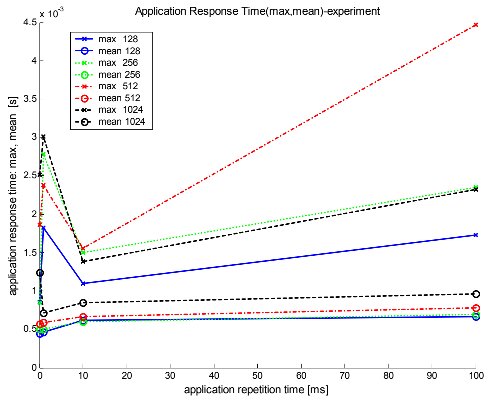

Figure 12 The application response time for different application repetition times

The application response time as a function of the application repetition time is depicted in the Figure 12. There is no significant dependency on application repetition time, which varies from 0ms to 100ms (0ms means as fast as possible).

The same conclusion can be done for different packet sizes, see Figure 13.The application response time grows up slowly with the growing packet size, because the communication time grows slowly too. Remember that Ethernet has huge throughput compared to the traffic generated in our experiments.

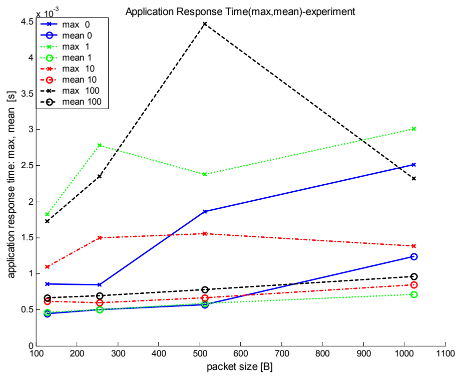

Figure 13 The application response time for different packet size

Figure 14 shows the mean application response time as a function of both the application repetition time and the packet size. Surprisingly, the application response time grows with prolongation of the application repetition time. This is due to the influence of the processor cache memory on the processing time.

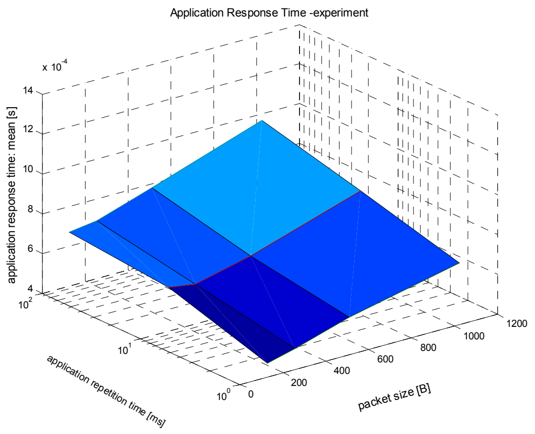

Figure 14 The application response time

5. Summary

This paper has described the experiments with ORTE (Open Real-Time Ethernet), an open-source implementation of the RTPS standard, referring to the documentation and the source code available at Sourceforge.

As a result we have we achieved the separation of the processing time and the communication time. The processing time gives higher jitter to the application response time. The communication time gives no jitter to the application response time due to the switched Ethernet.

Therefore, further development of ORTE will be focused on new implementations in order to obtain deterministic and possibly shorter processing time. The former can be achieved by using real-time operating systems (like RTAI or RT Linux), and the latter by using faster hardware.

6. Acknowledgement

This work was supported by the Ministry of Education of the Czech Republic under project LN00B096 and IST 35102 – OCERA. The OPNET Modeler is licensed under OPNET University Program.

References

[1] Czech Technical Univesity in Prague, Overall OpenSource project information containing ORTE, `http://sourceforge.net/projects/ocera/`, 2003

[2] L., Kleinrock, F.A., Tobagi, “Packet switching in radio Channels: Part I-Carrier Sense Multiple Access Modes and Their Throughput Delay Charakteristic”, *IEEE Transaction on Communication*, Corn-23, Dec., pp 1400-1416, 1975

[3] Real-Time Innovations, Inc., Real-Time Publish-Subscribe Wire Protocol Specification, Protocol Version 1.0, Draft doc, version 1.17. `http://www.rti.com/products/ndds/literature.html`, 2002

[4] Real-Time Innovations, Inc., NDDS Getting Started Guide, Version 3.0, 2002

[5] Real-Time Innovations, Inc, NDDS Data Delivery Performance, 2002

[6] S., Schneider, G.,Pardo-Castellote, M., Hamilton. “Can Ethernet Be Real Time?”, Real-Time Innovations, Inc., 2001

[7] P., Smolik, Z.,Sebek, Z.,Hanzalek, “ORTE - Open Source Implementation of Real-Time Publish-Subscribe protocol”, *2nd Intl Workshop on real-time LANs in the Internet age*, Polytechnic Institute of Porto, Portugal, 2003

[8] ORTE open-source code , `http://cvs.sourceforge.net/viewcvs.py/ocera/ocera/components/comm/eth/orte/`

[9] ORTE documentation, `http://www.ocera.org/archive/ctu/public/components/ethdev/ethdev-0.1.tgz`

[10] S., Sierla, “Middleware solutions for automation applications –case RTPS”, Diploma thesis, Helsinky University of Technology, 2003

[11] O.Dolejs, Z.Hanzalek, “Simulation of Ethernet for Real-Time Applications”, IEEE ICIT’03, December Maribor Slovenia, 2003