End-to-end Delay of Switched Ethernet Based on Time Division Multiple Access

Jieqiong Zhou
Capital University of Economics and Business, Beijing, China

***Abstract*—Focusing on the access control of terminal equipment base on time division multiple access (TDMA), an experimental network based on the RTNET protocol was established. The real-time Linux kernel was reconstructed and RTNET protocol stack of real-time Ethernet was transplant in the paper. Based on the time division multiple access mechanism, a number of tests were designed assigning different time slices. The results show that under the optimal allocation of time slices, the frame queuing delay in the switch is greatly reduced, and the end-to-end delay of Switched Ethernet applying RTNET protocol stack is significantly reduced compared to TCP / IP protocol.**

Keywords-end-to-end delay; time division multiple access; Switched Ethernet; allocation of time slices

I. INTRODUCTION

Research on real-time performance really is needed, to use Switched Ethernet to high real-time industrial control including train control communication. Real-time improvements for Switched Ethernet contain three aspects: (1) Design and optimization for topology [1]. (2) Research on the switch scheduling algorithm. (3) Research on the media access control mode of terminal equipment. This paper focuses on the use of time division multiple access (TDMA) of terminal equipment in Switched Ethernet.

RTNET, with key technology of TDMA, an improved real-time Linux operating system combining RTNET Ethernet protocol stack, is a software technology [2-3]. Experiments were completed to compare the real-time performance of operating system (OS) and network respectively in the Vxworks OS and Linux OS with RTAI/Xenomai transformation in literature [4], and results show that Vxworks OS has better real-time performance, but the real-time performance of network is less than RTNET. Literature [5] completed the transformation of Linux kernel based on Xenomai and transplant of RTNET on PowerPC platform, but analysis of TDMA cannot be found. Simulation of RTNET was established in literature [6], but there is no actual communication test.

Test segment of Switched Ethernet is built based on RTNET protocol in the paper. Optimization allocations of time slices are summarized according to test results.

II. RTNET PROTOCOL STACK

RTNET originally aimed to provide a hardware-independent real-time communication platform. RTNET real-time Ethernet protocol stack framework is shown in Figure1.

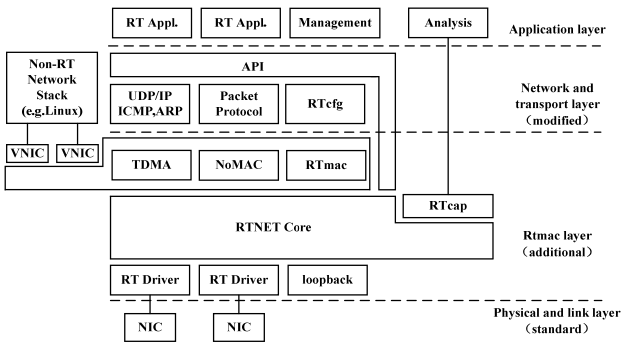


FIGURE I. FRAMEWORK OF THE RTNET REAL-TIME ETHERNET PROTOCOL

Real-time nature is mainly ensured on the basis of the following mechanisms:

(1) Real-time NIC driver. RTNET provides real-time driver for variety of popular Ethernet network interface, with hardware compatibility. NIC driver can achieve real-time link-layer protocol, but also retained the support of the standard TCP/IP protocol stack. Network initialization data is transmitted using TCP/IP protocol, while real-time data using real-time protocol.

(2) RTmac real-time layer. RTNET adds a real-time MAC sub layer between the data link layer and the network layer, called RTmac, using TDMA access mechanism avoiding conflict. Nodes in the RTNET network are divided into two categories-master node and slave node. At the beginning of each basic cycle, the two kind nodes correct their clock, and then each node sends data in its pre-assigned time slices.

(3) Improved UDP/IP protocol. RTNET improves the Address Resolution Protocol (ARP) from dynamic to static. In the initialization phase, the network identifies current nodes and generates a static routing table by RTcfg module. In addition, restructuring mechanism of fragmented IP datagram has been also optimized in RTNET.

(4) API interface. RTNET provides POSIX-compatible socket interface functions for user module and kernel, which facilitates the connection between applications and real-time network services.

III. RTNET TEST ENVIRONMENT

There are two steps to building a test environment for RTNET: single pc transplant for RTNET protocol stack and networking.

A. *Single PC Transplant for RTNET Protocol Stack*

RTNET transplant and started by the following steps:

(1) Linux kernel real-time improvements.

(2) RTNET protocol stack configuration, compile and install.

(3) Further configuration after installation.

Among them, the following principles are needed in the TDMA mode configuration:

(1) Time slice reservation for the clock synchronization. In initialization, each node identifies the network composition first, and then a static routing table is generated. The first time slice of basic cycle is always set aside out for the clock synchronization.

(2) Time slice reservation for the backup master node. The second time slice of each basic cycle is generally reserved for the backup master node to monitor the master node, so that when the master node fails, the standby master node can work as master node in time, and open the next basic cycle.

(3) Time slice configuration needs to set three properties: the offset (space between the starting time of the time slice and the starting time of the basic cycle), the length, and the belonging (the time slot belongs to which terminal equipment).

B. *Networking*

A test network segment is set up consists of four single PC, each of which nodes has been transplant with RTNET protocol stack and improved with Xenomai real-time Linux kernel. Specific hardware and software environment parameters for networking test are shown in Table 1.

TABLE I. ENVIRONMENT PARAMETERS OF NETWORK TEST

PC property	PC1	PC2	PC3	PC4
CPU model	Core2 P8400	Core-i3-2100	Core2 Q8300	Core2 Q8300
CPU clock speed	2.25GHz	3.10GHz	2.50GHz	2.50GHz
Memory size	4GB	4GB	4GB	2GB
NIC model	Intel 82567	Intel 82541	Intel 82541	Intel 82541
Link rate	100Mbps			
operation	Xubuntu			
Real-time Linux kernel	3.5.7-xenomai-2.6.2.1			
RTNET version	0.9.13			
Real-time NIC driver	rt_e1000e	rt_e1000	rt_e1000	rt_e1000
Node property	Maste node	Slave node A	Slave node B	Slave node C
IP address	136.137.138.20	136.137.138.21	136.137.138.22	136.137.138.23

IV. RTT TEST AND ANALYSIS

This paper analyzes the data end-to-end delay through the experiment of the network round-trip time (RTT). RTT test can reflect the processing capability of terminal device and real-time communication capability of the network segment.

Aiming at different time slice allocation methods performed three tests.

A. *The Master and Slave Node Time Slice are both 100µs.*

Time slice allocation is shown in Figure 2. The synchronization time slice is 100µs, and the master node send a request data frame to slave node A, and both the master and the slave node time slice are 100µs.

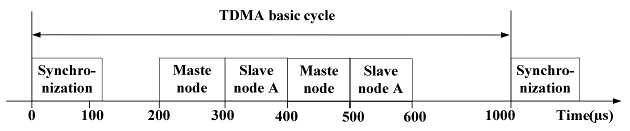


FIGURE II. BOTH THE TIME SLICES OF MASTER AND SLAVE NODE ARE 100MS

The master node sends respectively through TCP/IP protocol and RTNET protocol test data frames with length of 512 Byte and cycle of 10ms to the salve node A 5 times, and records 100 RTT values every time, which is shown in Table 2. Because the kernel clock accuracy in the Xenomai platform is improved, so the test results of RTNET stack values accurate to 0.1µs.

TABLE II. STATISTICAL RESULT WHEN BOTH THE TIME SLICES OF MASTER AND SLAVE NODE ARE 100MS

Protocol stack	Test NO.	Average value(µs)	Maximum value(µs)	Standard deviation(µs)
TCP/IP	1	612	3642	165
	2	622	7867	215
	3	678	5016	143
	4	679	4120	167
	5	681	4025	159
RTNET	1	1341.1	1412.9	60.5
	2	1635.5	1669.8	40.9
	3	1871.4	1825.9	36.1
	4	1978.2	1989.2	51.2
	5	1563.7	1597.6	48.5

The results show that: the average value of RTT measured in TCP/IP network can kept in 700µs or less, is smaller than in RTNET network, but the delay jitter in RTNET is smaller than in TCP/IP network, which standard deviation can kept in 70µs or less.

B. *The Master and Slave Node Time Slice are both 200µs*

Modifying the TDMA time slice allocation on the basis of test 1 shown in Figure 3, carry out the similar experiment with the same parameters, and the results are shown in Table 3.

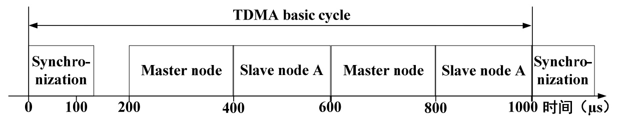


FIGURE III. BOTH THE TIME SLICES OF MASTER AND SLAVE NODE ARE 200MS

TABLE III. STATISTICAL RESULT WHEN BOTH THE TIME SLICES OF MASTER AND SLAVE NODE ARE 200MS

Protocol stack	Test NO.	Average value(μs)	Maximum value(μs)	Standard deviation(μs)
TCP/IP	1	674	4098	162
	2	675	4252	165
	3	659	4089	158
	4	621	7896	280
	5	623	3497	141
RTNET	1	459.5	469.2	14.1
	2	466.3	466.6	9.1
	3	392.9	401.2	11.5
	4	587.2	590.5	12.5
	5	531.2	536.5	12.6

The test results show that when the time slice of the master and slave node increases from 100μs to 200μs, the RTT values measured in RTNET network are greatly reduced, less than measured in TCP/IP network, and the delay jitters are also very small, and the standard deviations are even less than 15μs at the same time.

In order to analyze the reason that the average RTT values reduce with the time slice increase, the current widely used network analysis software Wireshark is applied, to capture the data frame of the master node in the paper, to help analyze the request and response. When using RTNET protocol stack for data transmission, data frame is divided into two categories. One is the synchronous frame, sent in broadcast by the master node in each initialization phase of the basic cycle. And the other one is the request and response data frame, sent between the master node and slave node A. When the time slice is set in 100μs, the master node sends a request data frame in its time slice (offset 200μs, width 100μs), but when the slave node A is ready to give a response to the request data, the time slice belong to it (offset300μs, width 100μs) has missed, so the response can only wait for its next time slice in the following basic cycle, leading to the increase of RTT values.

That is, when the time slices of the node are 100μs, the time waiting for the allocated slice leads to the larger RTT value, comparing to the 200μs time slice. Therefore, it's necessary to research the bandwidth allocation policy, which ensures the response data frame can be sent in the same basic cycle with the request data frame.

End-to-end delay in Switched Ethernet means that the time difference between the data sent from the source node to it received by the sink node, that in addition to the delay in the switch, also including the time in the source/sink node and the link. The specific component is shown in Figure 4.

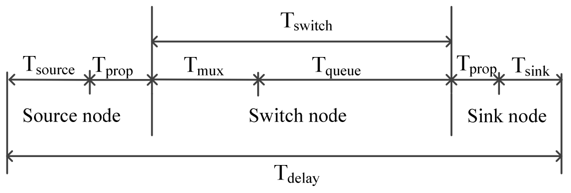


FIGURE IV. DELAY COMPONENT DIAGRAM OF FRAME TRANSMISSION IN THE SWITCHED ETHERNET

Each part of the delay including:

(a)Source node delay, T_{source} , which contains the time processing in the source node protocol stack, T_{source}^{proc} , the time waiting in the MAC layer buffer, T_{source}^{wait} , the time sending the data frame, T_{source}^{send} ,which related to the data length.

(b)Switch node delay, T_{switch} , which contains the basic delay T_{switch}^{basic} and the queuing delay T_{switch}^{queue} . This paper divides the T_{switch}^{queue} into waiting delay T_{switch}^{wait} and sending delay T_{switch}^{send} , which is the ratio of the data length and data transfer rate.

(c)Sink node delay, T_{sink} , which contains the time receiving data frame, $T_{\sin k}^{reci}$,which is the ratio of the data length and data transfer rate, and the time processing in the sink node protocol stack, $T_{\sin k}^{proc}$.

(d)Link propagation delay, T_{prop} , depending on the cable length between the communication nodes and the electrical signals transfer rate.

According to this, data frame end-to-end delay can be presented:

$$T_{delay}=T_{\text{source}}+T_{\text{switch}}+T_{\text{sink}}+T_{prop}$$
$$=\left(T_{source}^{proc}+T_{source}^{wait}+T_{source}^{send}\right)+\left(T_{switch}^{basic}+T_{switch}^{wait}+T_{switch}^{send}\right)+\left(T_{\sin k}^{reci}+T_{\sin k}^{proc}\right)+T_{prop} \quad (1)$$

The data frame queuing delay in the switch may be considered zero, for the Ethernet introducing TDMA mechanism. Assuming the time waiting for the corresponding time slice in the source node is T_{source}^{wait} , the end-to-end delay in the Switched Ethernet using TDMA according to (1) can be expressed:

$$T_{\text{delay}}^{\text{TDMA}}=T_{\text{source}}+T_{\text{switch}}+T_{\text{sink}}+T_{\text{prop}}$$
$$=(T_{\text{source}}^{\text{proc}}+T_{\text{source}}^{\text{wait}}+T_{\text{source}}^{\text{send}})+(T_{\text{switch}}^{\text{basic}}+T_{\text{switch}}^{\text{send}})+(T_{\text{sink}}^{\text{reci}}+T_{\text{sink}}^{\text{proc}})+T_{\text{prop}} \quad (2)$$

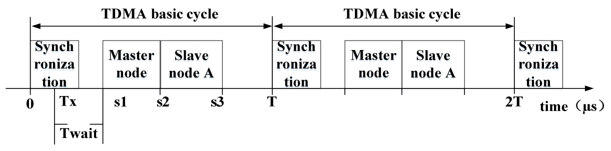


FIGURE V. TIME SLICE ALLOCATION OF RTNET

In general, the relationship between the waiting time of the request data $T_{source}^{req\_wait}$ and the timing sending it T_x, in the network which basic TDMA time slice allocated as Figure 5, can be presented:

$$T_{source}^{req\_wait} = \begin{cases} s_1 - T_x, 0 < T_x < s_1 \\ 0, s_1 \le T_x < s_2 \\ T - T_x + s_1, s_2 \le T_x < T \end{cases} \tag{3}$$

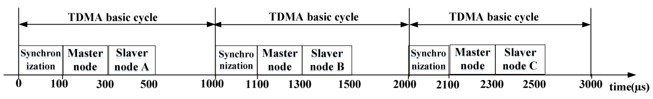
Assuming the slave node A can always respond the request data in the same basic cycle, then the relationship between the RTT and the timing sending the data T_x, can be presented:

$$\begin{aligned} RTT &= (T_{source}^{req\_\mathrm{wait}} + T_{delay}^{req}) + (T_{source}^{resp\_wait} + T_{delay}^{resp}) \\ &= \begin{cases} (s_1 - T_x) + (s_2 - s_1) + T_{delay}^{resp}, 0 < T_x < s_1 \\ T_{delay}^{req} + (s_2 - T_x) + T_{delay}^{resp}, s_1 \le T_x < s_2 \\ (T - T_x + s_1) + (s_2 - s_1) + T_{delay}^{resp}, s_2 \le T_x < T \end{cases} \end{aligned} \tag{4}$$

where, $T_{source}^{req\_\mathrm{wait}}$ and $T_{source}^{resp\_wait}$ respectively represents the waiting time of the master node sending request and the waiting time of the slave node sending response data, then T_{delay}^{req} and T_{delay}^{resp} respectively represents the end-to-end delay of request data and response data.

C. RTT Test in the Larger throughput Network

Master node sends requests to the slave node A, B, C successively, and the time slice allocated as Figure 6. The network throughput is about 12Mbps, with heavier load, where request data frames' cycle is 1ms and their size is 1500 Bytes. The RTT results in the TCP/IP protocol stack and the RTNET shown in Table 4.



TIME SLICE ALLOCATION IN LARGER THROUGHPUT NETWORK

TABLE IV. STATISTICAL RESULT IN LARGER THROUGH NETWORK

Protocol stack	Slave node.	Average value(μs)	Maximum value(μs)	Standard deviation(μs)
TCP/IP	Slave node A	1712	8682	231
	Slave node B	1668	7673	182
	Slave node C	2009	9854	203
RTNET	Slave node A	1235.7	1314.2	36.6
	Slave node B	1149.6	1163.5	10.2
	Slave node C	1258.9	1289.4	25.2

The results show that: with the increase of throughput in TCP/IP network, the average values of RTT increase significantly, and some test value even is greater than 2000μs, which has exceeded its deadline when assuming its deadline is equal to its transmission cycle, then the real-time performance of the network cannot be guaranteed. Whereas in the RTNET network, the average values of RTT and delay jitter both are reduced greatly and the maximum RTTs are all less than 1300μs, less than their deadline, then the real-time nature are guaranteed.

V. CONCLUSION

Summary, this paper describes an effective method for reducing the Switched Ethernet end-to-end delay and completes some typical test focusing on the time slice allocation of the nodes. The results show that with reasonable division of the time slice in the TDMA Switched Ethernet, the end-to-end delay can be reduced by 30% compared to the traditional TCP/IP network.

VI. ACKNOWLEDGMENT

This work was supported by the Beijing outstanding talent training funding (No. 2014000020124G119).

REFERENCES

[1] ZHANG SHUNYI, SUN LIHONG, SHU FEI, et al. design of the new algorithm based on genetic algorithm for Large-scale network topological[J].Journal on communications, 2001(6): 27-33.

[2] CARVAJAL G, FISCHMEISTER S. A TDMA Ethernet Switch for Dynamic Real-Time Communication[C]. 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Charlotte, 2010:119-126.

[3] HONG LU, HONG FENG, LI ZHENGBAO, et al. Efficient TDMA protocol of Underwater sensor network [J]. Journal on communications, 2012, 33(2):164-174.

[4] BARBALACE A, LUCHETTA A, MANDUCHI G, et al. Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a Hard Real-Time Application[J]. Nuclear Science, IEEE Transactions on, 2008,55(1):435-439.

[5] YUAN T, GUOGING R, QINZHANG W. Implementation of Real-time Network Extension on Embedded Linux[C]. 2009 International Conference on Communication Software and Networks, Macau, 2009:163-167.

[6] LING CHONGYU. Simulation and performance analysis applying OMNeT++ for Real-time Ethernet[D].Shang hai.Tongji University,2008.

[7] KISZKA J, WAGNER B. RTnet-a flexible hard real-time networking framework[C]. 2005 10th IEEE Conference on Emerging Technologies and Factory Automation, Catania, 2005:448-456.