International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 11 (2016) pp 7517-7522
© Research India Publications. http://www.ripublication.com

Token Ring Using Ethernet TRUE-Simulator

Nayel Al-Zubi
Department of Computer Engineering,
Al-Balqa' Applied University, 19117, Al-Salt, Jordan.

Abstract

Token Ring Using Ethernet (TRUE), protocol is a hyper protocol which manages standard protocols, Ethernet and Token Ring protocols. TRUE utilizes the traffic loads and activates the suitable protocol to manage the network. Ethernet protocol proves its performance at low and medium traffic networks where the collision rate is quite low comparing with high traffic networks. While Ethernet has its limitations in crowded networks, Token Ring protocol proves grantees the stability over different traffic level within the network. Also, Token Ring proves to overcome Ethernet at crowded networks because it is collision free protocol. TRUE uses the advantage of each protocol and activate it at the right time. This paper shows simulation work to compare the three protocols in term of the network delay and the benefit of using TRUE protocol in order to minimize network delay. In term of real implementation, TRUE is software protocol which requires no hardware changes or configuration and can be implemented on decentralized units and a large scale. The real implementation is out of the scope of this paper but highly encouraged as a future work.

Keywords: Ethernet, Token Ring, Network Protocol and Discrete Event Simulation

INTRODUCTION

Ethernet is one of the most commonly used protocols for bus topology. It is called sometime Carrier Sense Multiple Access/ Collision Detect, CSMA/CD, which senses the bus, i.e., carrier, before sending if it senses a signal on the bus, it will know that the bus is busy at the moment. But in case there is no signal on the bus at that time, the station will place its packet on the bus. The problem emerges when there are two or more hosts are trying to send at the same time on a free carrier or almost the same time. Then all signals will be collided. This causes what is called collision. To detect this collision the station needs to listen to the bus after it sent the packet directly for a certain period of time. If a station senses a distracted signal, i.e., a high voltage in a copper media, this will mean a collision just occurred [1-2]. If a collision is detected then they will follow a back-off algorithm to try to send the collided packet again later. The back-off algorithm is considered is battle-neck of the CSMA/CD protocol but at the same time it is powerful feature within bus topology [3-4].

Token Ring is usually used for the ring topology. This free collision protocol has a token which is the ticket that gives the host the right to use the media. There is only one token which means there is only one sender at a time. But this will be slow when there is low traffic because most traffic will be forwarded to exchange the token between hosts [5].

This paper suggests a hybrid solution which merges both algorithms within one protocol to minimize average delay. The proposed algorithm chooses between the two protocols to be run at the most effective situation based on the traffic level.

This paper is divided into four parts: TRUE protocol design, Simulation mechanism, Results and discussion and Conclusion.

TRUE PROTOCOL DESIGN

The new protocol, i.e., TRUE, is a hyper-protocol which manages two common protocols, Token ring and Ethernet protocols. Ethernet is an efficient protocol for light and average traffic network. The main source for the delay is the back-off algorithm which triggered after each collision and generates exponential delay for each host that involved in the collision. The average delay increases exponentially with the number of collisions and the probability of collisions increases with the number active hosts.

To overcome the Ethernet limitation in heavy traffic network, Token ring protocol is used. Token ring is used for ring topology, both types: logical and physical topologies. Token ring protocol manages transmission permission where one host and only one host can send at any time. Token ring manages permissions through a token which rotates between hosts and released if not required. Using token ring mechanism makes token ring protocol collision free protocol. Token ring protocol is more efficient than Ethernet for heavy traffic network. But in light and medium traffic network, Token rotation takes more time than the Ethernet collision delay.

TRUE protocol aims to deploy the most suitable protocol for traffic level.

TRUE protocol starts with Ethernet protocol and depends on the collision rate to detect live traffic load. If the collision rate exceeded the switching criteria, TRUE protocol broadcasts Ethernet-to-Token Ring packet from the most the host which detects the most collision rate that exceeds the switching criteria. Ethernet-to-Token Ring packet forces all hosts to follow token ring protocol with Token held by the host which issued the broadcast.

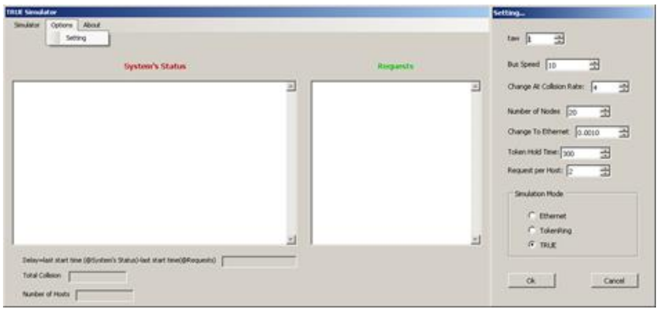
Similarly, switching back to the Ethernet protocol is done by checking the interval between token passes. If the period decreases and is stable then the network load is stable at low traffic rate which forces the host to issue Token Ring-to-Ethernet broadcast massage for all hosts.

TRUE utilizes both protocols by deploying them at most suitable environment. At consistent traffic network, TRUE stays invisible and doesn't affect the active protocol at all which makes the network identical to classical protocol.

DISCRETE EVENT SIMULATION

Discrete event simulator was built to simulate the three protocols, Ethernet, Token Ring and TRUE protocol. This simulator consists of three components: Ethernet bus simulation, Token Bus Simulation and Traffic behaviour monitor and protocol switching manager.

The simulator was built using C# programming language and with graphical user interface that allows the user to adjust number of settings, as shown in Figure 1.



Delay=last start time (@System's Status)-last start time(@Requests)
Total Collision
Number of Hosts

tau 1
Bus Speed 10
Change At Collision Rate: 4
Number of Nodes 20
Change To Ethernet 0.0010
Token Hold Time: 300
Request per Host: 2

Simulation Mode
○ Ethernet
○ TokenRing
● TRUE

Figure 1: The discrete simulator main GUI interface with a snapshot for the setting dialog

Ethernet Bus Simulation

This component of the simulator pops each request and calculates the sending time and the proposed ending time of that popped packet.

All packets are prescheduled and generated based on stochastic scenario that stimulates user behaviour in real life. The simulator defines media status based on Ethernet protocol rules and previous and current traffic. Figure 2 shows Ethernet packet's time frame which is used to identify collision and non-collision cases.

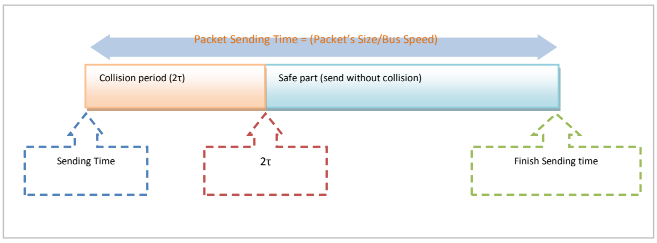

Packet Sending Time = (Packet's Size/Bus Speed)
Collision period (2τ)
Safe part (send without collision)
Sending Time
2τ
Finish Sending time

Figure 2: Ethernet packet's time frame identifying collision and no-collision cases.

One possible scenario is that the pervious request finished the utilization of the media this means the bus is free and the simulator can go ahead and send the current packet. Another scenario will be if the current request is trying to send before the previous request finished its sending and after the sensing time finished, this means the bus is busy right now. In this case the simulator selects a random delay for the current request to simulate user or program's behaviour which try to send the packet again after a random time –excluding aggressive users/programs-. The last possible scenario, when the sending time of the current request is within the collision detection interval for the previous request. In such case the collision detected, so a jam signal will be sent and all hosts which participate in this collision will follow the back-off algorithm where each host will select a random time based on its collision history to reduce the chance for another collision.

Within a crowded networks back-off algorithm dominate the network time and lead to a busy network with too many jam signals. While Ethernet protocol is quit useful and practical within low and medium traffic networks.

Token Bus Simulation

This type of protocol avoids the collision using reserved token signal. The token signal will give the right for a host to transmit and prevents other hosts from transmitting over the media. After the transmission process is finished for the token holder, it will release the token and the first host wants to transmit will pick it up. The Token Ring simulator will pop a new request from the requests list and there are two scenarios can be happened. First, if a host wants to transmit while another host is sending, it should wait and try on another time and competes to get the token. In another case if the host was lucky to pick the token and wants to send a data while the previous transmitter released the token on the Ethernet. Token Ring protocol is so efficient when there are a high number of senders over a certain period of time. Figure 3 shows the token ring packet's time frame format.

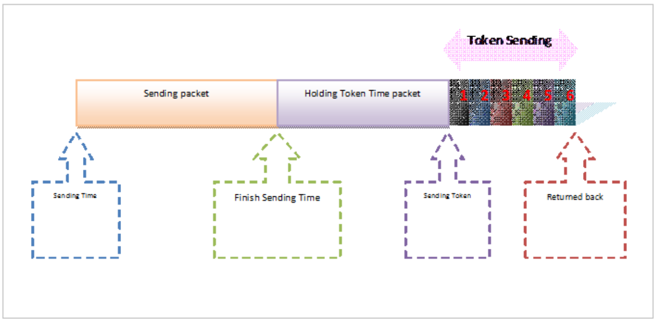


Figure 3: Token ring packet's time frame format

Token bus protocol proves an efficient utilization of the crowded networks were it is collision free protocol. Within low and medium traffic networks, token bus protocols waste network time to circulate the token which is more costly than the back-off algorithm in Ethernet algorithm.

Traffic Behaviour Monitor and Protocol Switching Manager

The traffic behaviour monitor and protocol switching manager is the main driver of the TRUE protocol. The traffic behaviour monitors the traffic level and based on the level of the traffic level it activates the most suitable protocol. The switching broadcast packet is not issued suddenly but based on an accumulated history of collisions in Ethernet protocol and token free rotation in Token bus protocol.

When the Ethernet protocol is activated, the manager monitors the network and counts the number of collisions within the network. As the number of collisions increases the delay of within the network time is increased.

While Token bus protocol is deployed, the traffic manager counts the number of free token exchanges.

Based on those two metrics, the protocol switching manager issues a switching broadcast packet for all hosts in order to deploy the most suitable protocol. Both metrics could be collected easily and will not affect the network behaviour since both are standard measurements.

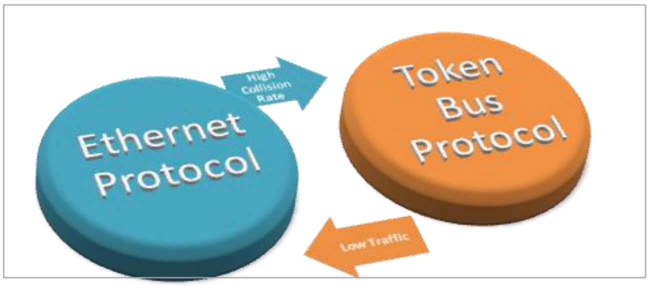
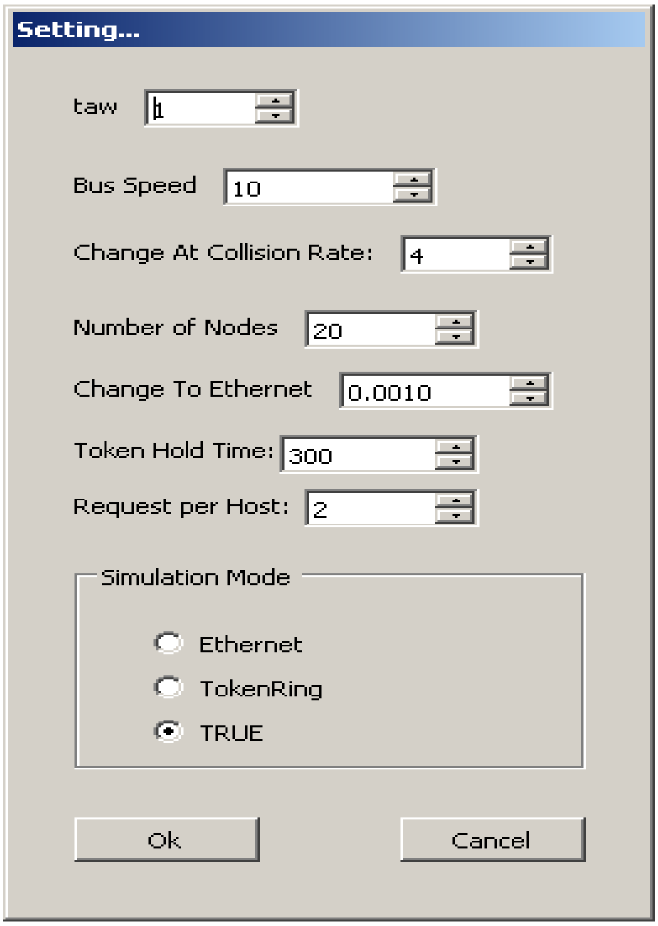


Figure 4: TRUE mechanism, switching between Ethernet and Token Ring protocols choosing appropriate one

Traffic behaviour monitor and protocol switching manager is suggested to be implemented within each host such as Ethernet protocol. This way of implementation is easier to be deployed and benefit from the decentralized feature where there is no sol-master device that has the right to control the protocol switching but any host can suggest switching the protocol based on its traffic observation.

Once a high collision is detected within the network, a jam signal will be issued by the first host that detects the high collision. Then a switching broadcast packet is issued and the token is owned by the host that detected the high collision rate. Then the network follows the token ring protocol. Switching back from the token bus to the Ethernet protocol decision is taken based on the free passes of the token within a period of time, i.e., unused token for certain period of time. Once the token utilization reaches low rates, a jam signal will be sent over the network and a switching packet is issued then the network will be back to a normal Ethernet protocol.

The simulation software simulates the network into three modes: Ethernet protocol, Token Bus protocol, and TRUE protocol. The simulation was run in the three modes to compare results. Number of parameters need to set in order to configure the simulation with a correct rate. Figure 5 shows the setting dialog where all configurable parameters are set.



taw: 1

Bus Speed: 10

Change At Collision Rate: 4

Number of Nodes: 20

Change To Ethernet: 0.0010

Token Hold Time: 300

Request per Host: 2

Simulation Mode

○ Ethernet

○ TokenRing

● TRUE

Figure 5: Setting window of the simulator, where simulation mode and parameters configuration are set

The parameters are:

I. τ (taw): is collision period where there is a possibility for collision to happen within a period of 2τ.

II. Bus Speed: is the maximum bus speed. For this simulation purposes, 10MByte per second was used.

III. Change at collision rate: is the switching threshold from Ethernet to token bus protocol.

IV. Number of Nodes: the number of involved hosts.

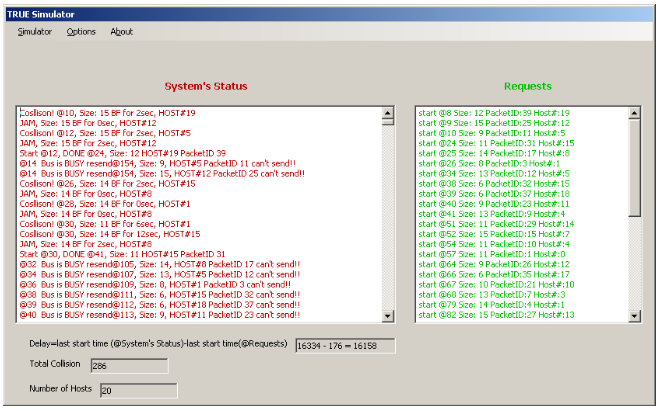
V. Change to Ethernet: is the fraction of time for token to be free in order to switch back to Ethernet protocol.

VI. Token Hold Time: is the available time for each host to hold the token.

VII. Request per Host: the number of requests per host to be simulated.

VIII. Simulation Mode: the type of protocol to be simulated, Ethernet, Token Ring or TRUE protocol.

The simulator shows the system status step by step through the simulation. So, it is easy to track back each event through the simulation time. Figure 6 shows a sample of the system status output, in red, and hosts requests, in green. Note that hosts requests were generated at random time with random packet sizes.



System's Status

```
Cosllison! @10, Size: 15 BF for 2sec, HOST#19
JAM, Size: 15 BF for 0sec, HOST#12
Cosllison! @12, Size: 15 BF for 2sec, HOST#5
JAM, Size: 15 BF for 2sec, HOST#12
Start @12, DONE @24, Size: 12 HOST#19 PacketID 39
@14  Bus is BUSY resend@154, Size: 9, HOST#5 PacketID 11 can't send!!
@14  Bus is BUSY resend@154, Size: 15, HOST#12 PacketID 25 can't send!!
Cosllison! @26, Size: 14 BF for 2sec, HOST#15
JAM, Size: 14 BF for 0sec, HOST#8
Cosllison! @28, Size: 14 BF for 0sec, HOST#1
JAM, Size: 14 BF for 0sec, HOST#8
Cosllison! @30, Size: 11 BF for 6sec, HOST#1
Cosllison! @30, Size: 14 BF for 12sec, HOST#15
JAM, Size: 14 BF for 2sec, HOST#8
Start @30, DONE @41, Size: 11 HOST#15 PacketID 31
@32  Bus is BUSY resend@105, Size: 14, HOST#8 PacketID 17 can't send!!
@34  Bus is BUSY resend@107, Size: 13, HOST#5 PacketID 12 can't send!!
@36  Bus is BUSY resend@109, Size: 8, HOST#1 PacketID 3 can't send!!
@38  Bus is BUSY resend@111, Size: 6, HOST#15 PacketID 32 can't send!!
@39  Bus is BUSY resend@112, Size: 6, HOST#18 PacketID 37 can't send!!
@40  Bus is BUSY resend@113, Size: 9, HOST#11 PacketID 23 can't send!!
```

Requests

```
start @8 Size: 12 PacketID:39 Host#:19
start @9 Size: 15 PacketID:25 Host#:12
start @10 Size: 9 PacketID:11 Host#:5
start @24 Size: 11 PacketID:31 Host#:15
start @25 Size: 14 PacketID:17 Host#:8
start @26 Size: 8 PacketID:3 Host#:1
start @34 Size: 13 PacketID:12 Host#:5
start @38 Size: 6 PacketID:32 Host#:15
start @39 Size: 6 PacketID:37 Host#:18
start @40 Size: 9 PacketID:23 Host#:11
start @41 Size: 13 PacketID:9 Host#:4
start @51 Size: 11 PacketID:29 Host#:14
start @52 Size: 15 PacketID:15 Host#:7
start @54 Size: 11 PacketID:10 Host#:4
start @57 Size: 11 PacketID:1 Host#:0
start @64 Size: 9 PacketID:26 Host#:12
start @66 Size: 6 PacketID:35 Host#:17
start @67 Size: 10 PacketID:21 Host#:10
start @68 Size: 13 PacketID:7 Host#:3
start @79 Size: 14 PacketID:4 Host#:1
start @82 Size: 15 PacketID:27 Host#:13
```

Delay=last start time (@System's Status)-last start time(@Requests): 16334 - 176 = 16158

Total Collision: 286

Number of Hosts: 20

Figure 6: Sample of the system status output, in red, and hosts requests, in green.

The detailed system status output as shown in figure 5 is used to investigate the system status event by event. But to draw a clear picture of the system behaviour the network load, number of requests per host, was changed automatically and accumulated delay was observed for the three modes.

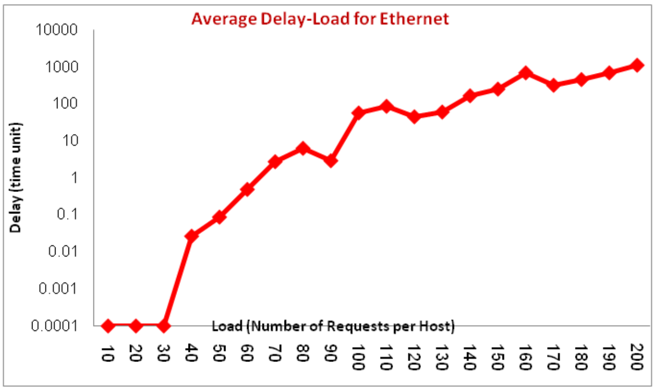


Figure 7: Ethernet average delay-load

As shown in figure 7, the delay increases with the load. This is expected behaviour for Ethernet where the number of collisions increases and the back-off algorithm causes exponential delay for the network time.

Our simulator was run on Token Bus protocol mode and generated the following result for delay.

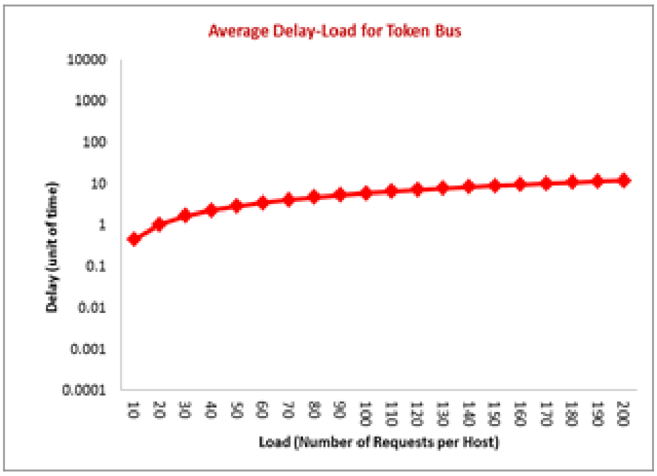


Figure 8: Token bus average delay-load

Figure 8 shows the stability of the average delay in the token bus. The network has almost a fixed rate of delay over different loads.

While the average delay in Ethernet networks is highly dependent on the load, Token Bus protocol makes it less dependent and more stable regardless the load. But Token Bus protocol pays the price in low traffic networks were the delay is stable and high comparing with the Ethernet. So, TRUE protocol comes to combine the advantages of both protocols and utilizes the right protocol at the suitable conditions.

The simulator was run on the TRUE mode and the result is shown in figure 9.

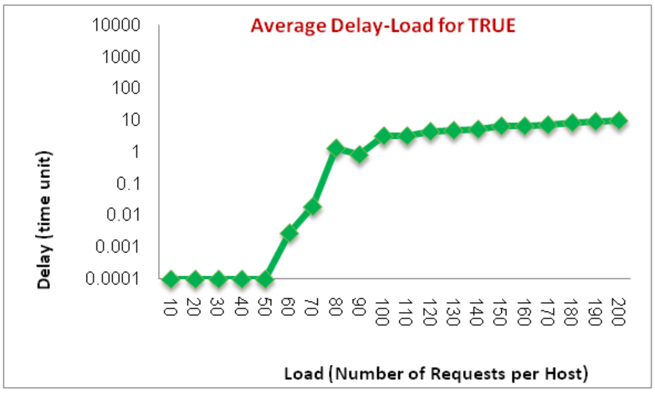


Figure 9: TRUE average delay-load

Figure 9 shows how TRUE protocol benefit from both protocols to minimize the delay. It is interesting to show that switching is not based on a certain number of requests but it could happen on the same number of requests but it depends on the real time traffic within the network. To explain this idea, we show the following table which shows the exact delay for each protocol at the same load.

Table I: Network Delay.

Load	Average Delay (time unit)		
	TRUE	Ethernet	Token Bus
60	28	4,858	34,419
80	12,822	61,707	46,410
100	31,601	559,503	58,425
120	44,371	435,022	70,419
140	52,057	1683,546	82,453

Table I show shows a sample of the delay within the network using each protocol. One could notice that the system is not just a fixed switching mechanism at certain load level but it is dynamic protocol switching based on real time traffic. For example at load 60 the delay in the network using TRUE protocol is much lower than the Ethernet itself because TRUE protocol switches back and forth between the two protocols Ethernet and Token Bus. This is caused because even with a fixed load at 60 requests per host, those requests are not uniformly distributed over time.

The average network delay enhancement ranges from the 0.5 to 0.005 of the delay in either network protocol comparing with TRUE protocol, depends on the network load.

CONCLUSION

TRUE protocol is dynamic protocol switching mechanism aims to activate the suitable protocol at most beneficial conditions. TRUE depends on two standard protocols, Ethernet and Token Bus protocols.

Simulation results show the benefit of using TRUE in speeding up the network response time which is critical and represent a challenge for network designers where they need to predict such a stochastic process.

No hardware changes are needed in order to deploy TRUE protocol and it could be install and work decentralized which will not affect the network in the future in case any of the hosts fail.

The simulation results shows that TRUE always overcome both standard protocols in low, medium and high traffic loads. The speeding up factor for TRUE protocol is the minimum at double of the standard protocol.

With such promising results, we highly recommend to deploy TRUE protocol for a lab network to test TRUE protocol performance practically and discover any practical implementation challenges.

REFERENCES

[1] P. Brenner, A Technicl Tutorial on the IEEE 802.11 Protpcol. BreezeCom.

[2] A.S. Tanenbaum. Computer Networks. Prentice-Hall, 4th edition edition, 2003.

[3] B. Smith, “Efficient and Accurate Ethernet Simulation”, unpublished.

[4] KG Shin SK Kweon and Q Zheng. (1999). Statistical real-time communication over Ethernet for manufacturing automation systems. In IEEE Real Time Technology and Applications Symposium DOI: 10.1109/RTTAS.1999.777672 (pp. 192-202)

[5] Stapleton, S. ; Sch. of Eng. Sci., Simon Fraser Univ., Burnaby, BC, Canada ; Hardy, R.H.S. ; Deen, M.J. ; Fortier, R. et al. (1989) “The Effect of Optimized Devices on the Performance of a Token Ring Network Interface”, in Conf. Rec. IEEE Int. Conf. Communications, Computers and Signal Processing, DOI: 10.1109/PACRIM.1989.48325