Priority Ethernets: Multimedia Support on Local Area Networks

Frank Adelstein and Mukesh Singhal
Dept. of Computer and Information Science
The Ohio State University
Columbus, OH 43210-1277

Abstract

This paper presents solutions to the problems associated with transfer of multimedia data on Ethernet and qualitatively study the performance of these solutions and their suitability to multimedia traffic. Specifically, priority protocols are developed that enable priority transmission of data on Ethernets. Multimedia data can be assigned priority over regular data, and the priority protocols can be used to deliver multimedia data over the Ethernet in a timely manner.

Key Words: Ethernet, multimedia traffic, priority.

1 Introduction and Motivations

Multimedia is becoming important in several domains [1, 2]. While the types of multimedia applications are diverse, they share a common underlying trait: "The multimedia data that is being exchanged from node-to-node is real-time in nature and needs to be delivered in a timely fashion. Multimedia data is isochronous, that is, it is sensitive to delays and requires a constant amount of bandwidth, as opposed to bursty traffic." Different media types of course have different tolerances, but they all have deadlines and data past its deadline can not be used [3]. Current approaches of transmitting regular data traffic which is bursty in nature and can tolerate a high variance in delay are not suitable for multimedia traffic which is isochronous. In multimedia systems, intelligent transmission of packets based on a data priority or deadline that will result in a much lower number of discarded packets is required [3].

Most business organizations and academic institutions already have LANs (local area networks) installed. However, as far as communication is concerned, the current Ethernets do not support high speed multimedia traffic [4, 5]. Current Ethernet protocols do not support real-time, isochronous traffic. Therefore, a practical solution is to add capabilities to the existing LANs in order to successfully transmit multimedia streams.

<u>State of the Art:</u> Ethernets are a baseband bus communication medium that use CSMA/CD [6]. A node listens to the channel before transmitting, to avoid collisions. When a collision occurs, each node waits for a random amount of time before trying again. This paradigm attempts to give all nodes an equal chance to transmit their data, while avoiding subsequent collisions due to repeated retransmission attempts [6]. There is no notion of priority at this level. Thus, transmission of multimedia data on an Ethernet does not get any preference and must compete equally with the transmission of regular data traffic.

Ethernet hardware, i.e., network interface, delivers only those packets to its node that are sent to it or broadcast to the entire net and ignores packets intended for other nodes. The network interface can be configured to be in "promiscuous mode" in which it receives all packets sent on the bus. However, this is likely to cause an unacceptable load on the CPU to process and discard all undesired packets.

Thus, Ethernet is designed to efficiently transfer regular data traffic which is bursty in nature and for which variable delay is acceptable. It is not suitable for multimedia traffic which requires constant delay, fast (or real time) delivery, and a large bandwidth. In this paper, we develop a set of protocols that enable priority transmission of data on an Ethernet. That is, if a set of nodes are trying to transmit packets, the node with the highest priority packet succeeds. By using a protocol for priority transmission, multimedia data can be given a higher priority than regular data and thus allowing multimedia data to be delivered in a timely fashion over the Ethernet.

Token ring networks support the concept of priority data transfer and suitability of a token ring for multimedia traffic has been studied by Amer et al. [7]. Priority Ethernet has been proposed in the context of traditional communication systems [8].

2 Priority Protocol for Ethernet

2.1 Issues Involved

The following are two main issues involved in the design of priority protocols of an Ethernet:

<u>I. Functional decomposition of the protocol:</u>
This deals with decomposing the protocol functions in components and identifying what functions the various components of the protocol must perform. There

are two distinct functions of the protocol.

1. *Dissemination of state information of nodes.* To decide which node should transmit its packet, each node must have knowledge of the highest priority packet of the other nodes on the Ethernet. Therefore, timely dissemination of this information is an important issue.
2. *Transmission of packets in priority order.* When a node wants to transmit a packet, in addition to the existing CSMA/CD checks, it compares the priority of its packet with that of the other nodes on the Ethernet. If it has the highest priority, it is allowed to transmit the packet immediately. Ties are broken by an *a priori* ranking of the nodes.

II. Structuring issues: These issues deal with assigning protocols functions to various layers (determining what layers are responsible for what). The concept of priority needs to be implemented at the lowest level possible, specifically, the Data Link Layer, which corresponds to the Ethernet device drivers.

Ideally, the network interface, the Ethernet hardware controller, should be "intelligent" and be able to implement a priority protocol on its own. However, if a hardware-oriented solution is not possible, priority protocol can be implemented in the device driver controlled by the node's operating system. An intelligent interface requires new, sophisticated hardware, and will be more expensive than the existing Ethernet controllers, while a device driver change only requires a software modification.

3 Techniques for Priority Transmission

After the priority information of packets at all nodes is disseminated, the rules for transmitting packets on Ethernets in the order of priority are relatively straightforward. However, dissemination of the priority information of nodes to all the nodes efficiently is a complex issue. Dissemination of priority information should take precedence over the transmission of data on an Ethernet. This is difficult to implement on the Ethernet that does not support priority communication to start with. We next present three methods for disseminating node priority information for the priority protocols on Ethernets.

Data Structures and Notations: The following data structures are used in all of the three methods. Note that the terms "site" and "node" are used interchangeably. Each node has two priority queues, called *priority* and *pending*. Each element in the *priority* queue at site i represents a site that wishes to send a message, and contains the site's id and the priority of the message it wants to send. Each element in the *pending* queue at site i represents a message that site i wants to send, and contains the priority of the message and a reference to the message. Elements in the queue are stored in priority order, with the elements with the highest priority at the front of the queue. For a full description of the techniques, refer to [9].

3.1 Synchronous Method

In the synchronous method to distributing the state information, after transmission of an arbitrary number of packets, there is a round of priority information exchange. In this round, each node sends a broadcast packet informing other nodes of the highest priority packet it wishes to transmit. Nodes broadcast their status information in an *a priori* fixed order. For example, node i broadcasts its information after it receives node $i-1$'s state information, after which node $i+1$ broadcasts its state. Provisions need to be made to handle adding and dropping nodes. This method should virtually eliminate Ethernet contention during the priority information exchange phase, however, some channel bandwidth is taken up by exchange of priority information and there is a processing overhead due to these broadcast packets moving up and down the protocol layers.

In this method, each node informs every other nodes of its highest priority message during the priority exchange phase. After that phase, every node has an identical copy of the *priority* queue and the node with the message at the front of the queue transmits its message and removes its entry from its *priority* queue. After having finished its transmission, that node informs the node with the message now at the front of its *priority* queue that it may now transmit its message. This continues until there are no entries left in the *priority* queue, at which point a priority exchange phase is initiated. If a node wants to send a message when its *priority* queue is empty, it initiates the priority exchange phase.

3.2 Modified Synchronous Method

The second approach is a variant of the first. The priority information exchange phase occurs after a fixed number of data messages are transmitted. However, rather than having a synchronous exchange of the priority information, nodes on the Ethernet transmit their priority messages in the priority messages exchange phase using CSMA/CD, with exponential backoff to handle collisions.

In addition, in this method if a site does not have any messages to send, then it does not transmit any priority information in the priority exchange phase, unlike the first approach in which a site must explicitly send a message indicating its queue is empty. However, since the number of sites which participate in the information exchange round vary, it is difficult to know when this phase has ended. The priority information exchange phase is considered ended if a predetermined fixed number of time units have elapsed since the last successful transmission of priority information. Therefore the duration of the period is dynamically adjusted based on current demand. This makes this approach highly desirable for heterogeneous traffic environments.

3.3 Asynchronous Method

The third approach is to allow asynchronous broadcast of node priority information. No distinct round of priority information exchange occurs. Instead, the broadcast of priority information is asynchronously embedded in the normal data transmission. That is, the priority data compete with the regular data for network bandwidth using the CSMA/CD protocol to resolve conflicts. In this approach, when a node gets a data packet for transmission from a user level process, it immediately broadcasts the priority of the packet if the packet has a priority greater than anything pending transmission at that node.

In this method, each node informs every other node of its highest priority message asynchronously, when the node wants to send the message. All sites receive the priority information messages since they are broadcast on the Ethernet, so all nodes add the new entry to their *priority* queue. The node with the highest priority message is the first to transmit its message. When it is finished transmitting, it informs the node with the next highest priority that it can begin transmitting its message. When a node receives this message, it deletes all entries ahead of it in its *priority* queue, since those messages already have been transmitted, and transmits its messages. And so on.

3.4 Completed Transmission

Recall that in order for the second highest priority node to begin transmitting, it must be informed when the highest priority node has finished transmitting its packet. We have identified two approaches to disseminating the information that a node has finished transmitting its data. In the first approach, every node monitors the Ethernet and updates their *priority* queue every time a node successfully transmits data. This requires current Ethernet hardware to be set in "promiscuous mode" which may put a heavy load on the CPU. However, if the interface were intelligent and could update the queue without the CPU having to process all the packets, this can be a viable approach. In the second approach a node, after successfully transmitting data, transmits a packet indicating that it finished transmitting data.

This problem has been solved in the previous three methods with a message that is sent point-to-point to the node with the next highest priority message. The other nodes on the Ethernet will not have current information as the the highest priority node, but every since every site starts with an identical copy of the *priority* queue, all sites have the same well defined order of transmission, and know which site should transmit when they complete their transmission. For all three methods, there is an overhead of a packet that is transmitted after every data message. The more packets per data message, the smaller the overhead of the broadcast packets. However, in the worst case when each data message occupies one physical Ethernet packet or frame, the number of messages transmitted on the Ethernet are doubled.

3.5 Qualitative Analysis of the Methods

Efficiency: The synchronous approach can suffer from delays during the information exchange phase. Since each site must wait until its turn to broadcast its priority information, every delay incurred by each site adds to the time required to complete the entire exchange phase. In addition sites that have no message to send must still send a priority information message. In high traffic when most sites have message to send, this method is very efficient.

The second approach alleviates these delays by allowing the network interface to transmit the priority information as soon as the Ethernet is free during the priority information exchange phase. However, delays can be incurred due to packet collisions and retransmissions during this phase. In addition, a node must wait for a timeout period before the information exchange phase ends. If the timeout value is too small, then some sites will not be able to transmit their priority information and therefore not be able to transmit their data in the following data transmission phase. If the timeout value is too large, then time is wasted waiting for the information exchange phase to end. The second method is superior to the first in heterogeneous traffic, where few sites transmit messages at a higher rate than others. If all sites do have a message to send in every phase, then the second method is no more efficient than the first.

For low traffic, the third method is better, since a high priority message does not have to wait for the priority information exchange phase before it can be transmitted. In the other two methods, if a high priority message arrives at a node to be transmitted just after a priority exchange phase, it must wait for all of the messages to be transmitted before the its priority information can be exchanged with the other sites. In this regard, the third method is the most fair, however it is also prone to the most collisions, which are discussed subsequently.

Potential collisions: Collisions directly relate to wasted bandwidth on an Ethernet, and should be minimized. The first method should be free from collisions since the priority information exchange phase is completely synchronous and only the site with the highest priority message will transmit during the data transmission phase. However, there will be a lower Ethernet bandwidth utilization, since there will be "dead time" on the net during the delays incurred by processing the priority information messages. The only way a collision can occur is if two sites simultaneously send a message requesting priority after the system has been idle. The second method should reduce these delays by allowing asynchronous broadcast of the priority information messages during the priority information exchange phase, although delays due to collisions are possible. During actual data transmission, there should be no collisions, since only the highest priority site on the net will attempt to transmit its data. The third method can suffer from more collisions and subsequently delays due to collisions than the other two methods since priority information messages can be broadcast at any time. However, this method will not have the time lost due to the state information exchange rounds that the first two methods have.

Potential improvements from using a smart interface: A smart interface is capable of managing the local *priority* queue and updating it appropriately when it receives priority information and request broadcast messages, without intervention from the node's CPU. The first method benefits the most, since the synchronous operations of the priority information exchange phase consume CPU resources.

For every priority information message received, each node's CPU must decide if it is its turn to send the node's priority now or not. Every priority information message must go up and down the protocol layers before it can be processed. In addition, each message generates context switches which consume additional CPU cycles. With a smart interface, all the details of when to broadcast its priority information are handled by the smart interface, which would already be tracking which sites have already broadcast their priority information and if it is this node's turn to broadcast its priority now or not.

The other benefit of having a smart interface would be shared by all three methods. That is, time is saved that would be needed to process the broadcast messages, both the time spent for context switches (at least two per message) and the CPU time needed to process the broadcast message itself. This time would be proportional to the number of sites on the local Ethernet, since each site must broadcast its priority. In addition, the network interface would be able to process the messages sent when a site finishes transmitting its data.

Fairness: The priority information exchange phases used by the first two methods can be invoked either after every message, or after some fixed number of messages. This parameter will affect the fairness versus starvation issue of the algorithm. If the value of the parameter is low, then the priority exchange phases occur more often, so it is more likely that a site with a high priority message can send it sooner. However, other sites, with lower priority message might face starvation. If the parameter is larger, then the information exchange phases occur less often and more of the sites will eventually be able to send their messages, however a site with a higher priority message will be forced to wait until after the next information exchange phase until it can transmit its data. A reasonable compromise is to set the parameter to the number of sites that want to send a message at the priority exchange phase.

Message overhead: All three methods should have a comparable message overhead, since one control message is required to establish the priority and one is required to relinquish it, although the collision rate differs for each method. The methods only differ in the way they disseminate that information.

4 Conclusions

Traditional protocols for local area networks, e.g., an Ethernet, were designed to handle data traffic which is bursty in nature and for which variable delay is acceptable. These protocols are not suitable for multimedia traffic which requires pretty much a constant delay in packet deliveries as well as fast delivery of packets.

In this paper we presented three protocols for priority transmission of data on Ethernets: (i) the synchronous method, which virtually eliminates packet collisions by having a well-defined order for data transmission that is established during the priority information exchange phase in which nodes have a fixed *a priori* order to transmit their priority data, (ii) the modified synchronous method, which relaxes the ordering constraint during the priority exchange phase to improve efficiency for heterogeneous traffic, and (iii) the asynchronous method, which eliminates the priority information exchange phase to reduce the time high priority messages wait until they can be transmitted. By giving multimedia data priority over regular data, these priority protocols will allow a higher throughput for multimedia data with fewer packets dropped (due to missed deadlines).

Besides multimedia applications, these protocols can be used in applications (e.g., control systems, industrial process control, automated manufacturing, air-traffic control, program trading, etc.) where certain tasks must meet deadlines. Critical tasks can be assigned higher priority to ensure better service to them. We identified the functions of new (intelligent) network interfaces that will perform most of the priority transmission related chores in hardware. And finally, we presented a qualitative analysis comparing the three methods of disseminating priority information based on several criteria. Future research involves a simulation study of the three techniques.

References

[1] A.D. Narasimhalu and S. Christodoulakis. Multimedia Information Systems: The Unfolding Reality (Guest Editors' Introduction). *Computer*, 24(10):6 – 8, October 1991.

[2] Special issue on multimedia systems. *IEEE Journal On Selected Areas In Communicaions*, 1990.

[3] Domenico Ferrari. Client Requirements for Real-Time Communication Services. *IEEE Communications Magazine*, pages 65 – 72, November 1990.

[4] Dave Andrews. High Stakes for High-Speed Ethernet. *Byte*, pages 22 – 23, October 1993.

[5] Daniel Minoli. Isochronous Ethernet: Poised for Launch. *Network Computing*, pages 156 – 161, August 1993.

[6] R. Metcalfe and D. Boggs. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM*, 19(7):395 – 404, July 1976.

[7] Khaled Amer, Ken Christiansen, Tora Toher. Multimedia Applications on IBM 16-bit Token-Ring LANs. In *IBM Personal Systems Journal*, April 1993.

[8] Fouad A. Tobagi. Carrier sense multiple access with message-based priority function. *IEEE Transactions on Communications*, COM-30:185 – 200, January 1982.

[9] Frank Adelstein and Mukesh Singhal. Priority ethernets. Technical report, The Ohio State University, July 1994.