SUPPORTING MULTIMEDIA COMMUNICATION OVER A GIGABIT ETHERNET NETWORK

B. Daines,* J.C.L. Liu,** and K. Sivalingam***

Abstract

The need for gigabit networks is driven by the explosive growth of the Internet and the World Wide Web. Applications such as streaming multimedia, virtual reality, high-performance distributed computing, and distance learning will all drive the bandwidth requirements into the multi-gigabit range. This article introduces our joint effort to support multimedia communication using a set of gigabit Ethernet network components. It first describes the hardware/hardware design on the gigabit network interface card, a buffered hub, then a gigabit routing switch for providing the backbone. The switch can route packets based on layer 2 MAC addresses, layer 3 (IP or IPX) addresses, and layer 4 and above policy-based schemes. The design principles common to these different components are: (1) simplicity, to ensure the network components reach the performance beyond the gigabit speed, and (2) end-to-end solutions, to eventually integrate the gigabit component into the emerging gigabit network. With the accomplished network hardware/hardware, we bring in our unique multimedia delivery software to investigate the support for multimedia applications over the gigabit Ethernet network. The main challenges of retrieving streaming videos are large data size, real-time constraint, and supporting concurrent accesses. Performance measurements using netperf benchmark, our own TCP-based benchmark, and video server applications are presented in the article. The results indicate a promising result with maximum throughput of 190-200 Mbps achieved at the TCP level using Linux (even on low-end Pentium machines). With faster PCI busses and more powerful processors such as UltraSPARC and DEC Alpha, higher TCP throughput in the range of 400 to 500 Mbps was achieved. The video server experiments indicated that eight to nine simultaneous video streams at 16 Mbps each can be handled by the server.

1. Introduction

The motivation for gigabit network speeds can hardly be overstated. The need for gigabit networks is driven by the explosive growth of the Internet and the World Wide Web. There have been a number of gigabit research testbeds built in the past few years, some of which are described in [1]. Gigabit networks are beginning to leave the research domain and enter the enterprise network domain. The national backbone network speeds are ranging between 155 Mbps and 622 Mbps, as in the case of vBNS [2]. It will not be long before the wide-area backbone networks move to gigabit speeds. The increasing multimedia content of web traffic, among others reasons, will ensure that the gigabit bandwidth is significantly utilized.

Local area network bandwidth is following the same trend. Applications such as streaming multimedia, virtual reality, high-performance distributed computing, and distance learning will all drive the bandwidth requirements into the multi-gigabit range. For instance, a video server can generate 2.88 Gbps traffic from the storage subsystem [3]. In addition, the nature of traffic distribution at the enterprise network has changed. Traditionally, 80% of the traffic traversed local subnets, and only 20% traversed the enterprise backbone. Now the situation has reversed, with an increasing percentage of traffic traversing the backbone. It is essential to provide gigabit speeds to both the individual workgroups and subnets, and the enterprise backbone.

This article introduces our joint effort to support multimedia communication using a set of gigabit Ethernet network components. It first describes the hardware/hardware design on the gigabit network interface card, a buffered hub, then a gigabit routing switch for providing the backbone. These three hardware/hardware components are critical for the gigabit Ethernet deployment. We foresee that the first step is the deployment of small "power" or pilot workgroups within a local area network (LAN). This requires the development of gigabit interface cards, point-to-point optical links, and a gigabit hub. Examples of such workgroups include specialized research laboratories (like ours). Therefore, the development of multimedia and high-speed file server environment can be provided within the LAN. The next step is the integration of these servers through gigabit switches and aggregation of these gigabit switches. Thus, the deployment of gigabit capability in the building and campus backbones (i.e., MAN) can be
accomplished.¹

There are many design issues and challenges in the hardware/firmware level to achieve high-performance Ethernet that can perform beyond the gigabit speeds. We present the common challenges faced in MAC level design, among these different network components. Gigabit network interface cards (GNIC) is presented in Section 2.2. The particular design issues that we discussed include (1) 802.3 frame compatibility, (2) design challenge of the MAC ASIC to operate the GNIC at the line speed, and (3) techniques for reducing host CPU utilization with descriptor-based direct memory access (DMA) and interrupt filtering.

Section 2.3 presents the gigabit hubs that will scale up the number of hosts that can communicate with each other. Unique design features of the buffered gigabit hub include (1) full-duplex for eliminating the CSMA/CD collision issues, (2) congestion control to avoid frame dropping, and (3) round-robin scheduling to prevent “packet clumping.” The design of a gigabit routing switch is then presented in Section 2.4 with design descriptions on (1) architecture issues, (2) novel parallel access shared memory architecture, and (3) priority queue design. The switch can route packets based on layer 2 MAC addresses, layer 3 (IP or IPX) addresses, and layer 4 and above policy-based schemes.

The common design principles across these different components are (1) simplicity, to ensure the network components reaches the performance beyond the gigabit speed, and (2) end-to-end solutions, to eventually integrate the gigabit component into the emerging gigabit network. We extend the design considerations for supporting emerging distributed multimedia applications. The main characteristics of supporting concurrent streaming videos are introduced. Then the multimedia-friendly features associated with the different network components are described. Related new IEEE draft standards (e.g., 802.1p and 802.1Q) are introduced as an ongoing investigation. Possible approaches are considered in port-level and box-level to eventually ensure an end-to-end solution for distributed multimedia applications.

Gigabit Ethernet testbeds have been established at University of Florida and Washington State University. One Pentium Pro and one Pentium II computer, each equipped with a GNIC card, are connected via an optical link. A testbed connecting up to four clients and a server using the gigabit hub (FDR) has been set up. Performance measurements using netperf [4] benchmark, [5] benchmark, our own TCP-based benchmark, and video server applications are presented. The results indicate that a maximum throughput of 200 Mbps was achieved at the TCP level using Linux on Pentium. With faster PCI buses and more powerful processors such as UltraSPARC and DEC Alpha, higher throughput in the range of 400 to 500 Mbps was achieved. The video server experiments indicated that eight to nine high-quality streaming videos at 16 Mbps each can be handled by the server over the gigabit Ethernet network.

¹ The University of Florida has recently adopted the gigabit Ethernet as the campus backbone.

2. Hardware/Firmware Design Considerations

We begin this section by addressing network interfaces required to support gigabit speeds. Before introducing the network interface design, it is first necessary to describe the IEEE 802.3z standard.

2.1 IEEE 802.3z Standard

The IEEE 802.3z committee made the standard the MAC and PHY (physical) layers for Gigabit Ethernet. The PHY interface is defined for fiber operating at 1000 Mbps—1000 BASE-SX and 1000 BASE-LX. The 1000 BASE-SX standard operates at 800 nm, uses multi-mode fiber [6], and is defined for a maximum link length of 260m at 62.5 µm and 440m at 50 µm core diameter. The 1000 BASE-LX standard for single-mode fiber [6] operates at 1300 nm and can traverse a maximum link length of 3Km. Typically, single-mode fiber is more expensive than multi-mode fiber. In addition, a copper-based physical interface called 1000 BASE-CX using twinax cable has been defined for a distance of 25m. The MAC frame format is shown in Fig. 1.

Figure 1. IEEE 802.3z medium access control frame format.

The frame format defined for 1 Gbps Ethernet is identical to that defined in IEEE 802.3 for 10 Mbps Ethernet and IEEE 802.3u for 100 Mbps Ethernet. From a timing perspective, the Inter Frame Gap, which is the equivalent of 16 bytes, is reduced to 0.16µs, compared to 1.6µs in 100 Mbps Ethernet and 16 µs in 10 Mbps Ethernet. This places a stricter requirement on synchronization and preamble collision.

The frame consists of the following fields:

DA, SA: The destination and sender fields, each represented using the IEEE 802.3 48-bit MAC address format.
LEN: The length of the data field, in bytes—2 bytes allocated for this field.
DATA: Variable length data field with a minimum of 46 bytes and a maximum of 1500 bytes.
FCS: Frame check sum requiring 4 bytes.

With the common frame standard introduced, we will introduce the detailed design for the gigabit GNIC in the next subsection.
2.2 Gigabit Network Interface Card

This subsection describes the Gigabit Network Interface Card (GNIC) we have developed. The original GNIC is designed to operate with the 64-bit, 33-Mhz PCI bus and the SBus from SUN Microsystems. The GNIC is also compatible with current 32-bit, 33-Mhz PCI buses. The next generation card, denoted GNIC-II, is designed to operate with 64-bit, 66-Mhz PCI buses such as those available on Ultra-30 workstations from SUN Microsystems.

The theoretical bandwidth of a 32-bit, 33-Mhz PCI bus is about 1 Gbps; practical limits are around 800 Mbps. Similarly, the theoretical and practical limits of the 64-bit, 66 Mhz PCI bus are around 4 Gbps and 3 Gbps, respectively. This is an indication that high-end systems are currently capable of handling gigabit speeds. As processors and buses get more MIPS and bandwidth, more systems can avail themselves of the network's gigabit speeds.\(^2\)

The GNIC consists of an application-specific integrated circuit (ASIC) chip, packet buffer memory (512 KBytes), serializer/deserializer chip, and physical layer components. One important feature of the GNIC is that its data-path is designed to operate at line speeds to utilize the potential gigabit bandwidths from the 32-bit and 64-bit PCI buses.

![Block diagram of a gigabit network interface card](image)

Figure 2. Block diagram of a gigabit network interface card.

The block diagram of the ASIC chip is shown in Fig. 2. The ASIC chip has a PCI bus interface to the host bus. On the other side of the bus interface, a pair of DMA (direct memory access) controllers—one each for transmit and receive—are provided to reduce host CPU utilization. A dual-burst FIFO of 512 bytes is provided—one each for Transmit and Receive. This memory is interconnected to the on-card memory through an external FIFO interface. The GMAC ASIC block implements the IEEE 802.3x standard and the 802.3x full-duplex flow control. The GMAC is a single entity and is used on a number of other gigabit interface cards. The GMAC is connected to the serializer/deserializer, the fiber transceiver, and the physical interface (not shown in figure).

One of the main design objectives was to reduce the host CPU utilization, and a number of optimizing techniques have been used. Some of them are described below:

- The PCI Bus Master in a PCI 32- or 64-bit bus operating at 33 Mhz accesses host memory directly to offload the host CPU. On-board Independent Receive (RX) and Transmit (TX) Descriptor-Based DMA Processors (DBDMA) reduce host utilization by streaming RX and TX data to and from host memory without host intervention. A descriptor is similar to a CPU instruction that is executed by the DMA processors.
- Transmit chaining reduces host CPU utilization by transferring an arbitrary number of packets from host memory to the GNIC without host intervention.
- The GNIC will not interrupt the host when the host is currently servicing packets for transmission or reception. This allows the host to adapt the interrupt rate to network load.

An enhanced data path is provided on the ASIC to achieve gigabit speeds. The dual-burst FIFO—one each for RX and TX—allows simultaneous transfer of data from the host memory to the PCI FIFOs and from the PCI FIFOs to the wire (and vice versa). Also, bursting multiple descriptors reduces PCI overhead, as the PCI protocol overhead is incurred each time the master arbitrates the bus. Hence, the more data that are burst over the bus for each arbitration, the higher the efficiency of bus bandwidth usage.

The provision of large packet buffers on the card, of size between 512 KB and 2 MB, reduces the probability of dropped packets during extended 1 Gbps bursts. This buffer is partitioned for RX and TX purposes. For example, when the buffer size is 512 MB, it is partitioned so that the RX buffer varies from a minimum of 480 KB to a maximum of 512 KB. On the other hand, the TX buffer is limited to a maximum of 32 KB. The rationale is to accommodate bursts during reception without dropping packets.

We have observed a sustained throughput up to 800 Mbps achieved in the hardware/hardware level using this GNIC (excluding the protocol stack and operating system overhead). Without our unique design, a network interface card for gigabit communication might occupy all the host CPU utilization, making it unfeasible to achieve the sustained 800 Mbps throughput. With our GNIC design, the efficiency of the CPU utilization has been improved 6.8 times compared to conventional Ethernet network interface cards.

2.3 Full Duplex Repeater for LAN

Interconnection of more than one GNIC card is accomplished by the Full Duplex Repeater (FDR) considered in this subsection. As the FDR provides the MAC functionality, we first discuss the current Ethernet environment, and a solution to provide medium access at gigabit speeds.

In most current 10/100 Mbps Ethernet networks, the network interface cards (NICs) are connected to each other via a hub. The network uses the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) mechanism to
Table 1
Comparison of Key Features of Typical Hub, FDR, and Switch

<table>
<thead>
<tr>
<th>Feature</th>
<th>Hub</th>
<th>FDR</th>
<th>Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Full/Half-Duplex</td>
<td>Half-Duplex</td>
<td>Full-Duplex</td>
<td>Full-Duplex</td>
</tr>
<tr>
<td>Network Diameter</td>
<td>200 m (CSMA/CD dependent)</td>
<td>PHY dependent</td>
<td>PHY dependent</td>
</tr>
<tr>
<td>Performance</td>
<td>&lt; 1000 Mbps</td>
<td>1000 Mbps</td>
<td>&gt; 1000 Mbps</td>
</tr>
<tr>
<td>CSMA/CD Collisions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MAC Checks Frames</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Store &amp; Forward Buffers</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Congestion Control</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fair Bandwidth Allocation</td>
<td>No</td>
<td>Yes</td>
<td>Scheduling Alg. Dependent</td>
</tr>
<tr>
<td>COST</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

provide shared access to the medium. The performance of CSMA/CD is highly dependent on the ratio of end-to-end propagation delay (time for electro-magnetic waves to travel in the medium) to the average packet transmission time. Let us denote this ratio as $a$. The higher the value of $a$, lower the efficiency and overall utilization of the network, and vice versa [7]. The performance dependence on $a$ occurs mainly because of the time required to sense carrier and detect collisions.

Let us consider a 10 Mbps Ethernet with a 2 Km span and 64-byte minimum packet length. The ratio $a \approx 0.2$ does not cause significant performance degradation. However, when the speed is increased to 1 Gbps, the value $a = 20$ for the same distance span, which significantly decreases performance [7]. Thus, increasing transmission speed by itself does not necessarily imply higher performance. There are two ways to combat the performance degradation: increase the minimum packet length, or decrease the distance span.

As the IEEE 802.3z standard maintains the minimum packet length of 64 bytes, the network distance has to be reduced to 200 m to maintain the lower value of $a$. A technique called virtual collisions has been proposed [8] that increases the network distance to 400 m. For a typical hub, multiple transmissions arriving simultaneously results in all packets being lost. In the virtual collision solution, the hub is able to correctly broadcast exactly one of the collided packets. Thus, despite collision, one packet can be successfully transmitted. An alternative to the CSMA/CD shared medium architecture is to use a switched medium such as Switched Ethernet. Although switching significantly improves performance, it is associated with higher costs. Therefore, we prefer an intermediate between fully shared and fully switched solution.

The Full Duplex Repeater (FDR) uses a distinct approach to overcome the limitations of CSMA/CD and at the same time achieve better utilization and fairness of service. The term repeater is actually a misnomer, as the FDR operates at the data-link layer and processes MAC and link-level frames. A repeater, however, operates at the physical layer and is not aware of link-level frames. The FDR architecture combines switching and shared design concepts to achieve switch-like performance while maintaining the cost and ease-of-use advantages of a shared hub. The FDR provides maximum throughput with full-duplex ports, collisionless frame forwarding, and congestion control. A fair scheduling algorithm (currently round-robin) is used to fairly allocate forwarding bandwidth to all the ports.

A comparison shown in Table 1 highlights the differences between a typical hub, the FDR, and a switch.

The FDR architecture, illustrated in Fig. 3, consists of multiple ports based on the IEEE 802.3z Gigabit Ethernet standard, a 1000 Mbps forwarding bus, and a frame forwarder. Each of the ports includes a 802.3z physical layer device (PHY), an 802.3z media access controller (MAC), and buffers for incoming and outgoing frames, and uses a 802.3x flow-control logic. The forwarding bus is controlled by a frame forwarder.

Figure 3. Architecture of the full duplex repeater.

To illustrate how frames travel through an FDR, Fig. 4 shows the logical flow of frames passing through the physical architecture: input, forwarding path, and output.

An incoming frame enters through Port 1. During the input stage, the frame passes through the PHY and the MAC, and is then queued in the buffer on Port 1, which is controlled by the frame forwarder. The frame then moves into the forwarding path stage, when the frame forwarder selects the buffer on Port 1. The frame forwarder then con-
Figure 4. An example of frame passing in the full duplex repeater.

This port for transmission onto the bus and configures all other ports to listen to the bus. The result is that the selected port transmits this frame (and only this frame) onto the bus; all of the other ports get a copy. Then, during the output stage, the frame that entered Port 1 will exit the FDR on the second and third ports as illustrated in Fig. 4.

2.3.1 Input

This section describes the internal details of input section of the FDR.

**Full-Duplex 802.3z MACPHY:** Every frame flowing through an FDR passes through the PHY and MAC prior to being queued in a buffer for forwarding. The PHY converts the data signal on the wire or optical fiber into a digital data stream for the MAC process. The MAC processes this digital data at the frame level to make sure that the frame is valid and has no errors.

**Congestion Control (802.3z Flow-Control):** Each of the ports can provide traffic at a rate that matches the maximum rate of the shared bus. However, even when the offered load exceeds the bandwidth of the shared bus, the FDR will not drop frames because FDR technology supports congestion control. With the recent standardization of 802.3x full-duplex control, an FDR port can signal to the end-system that its buffer cannot accommodate more data. The end-system interprets this message and stops transmitting until the FDR port buffer can accept more data.

For example, let us assume that two or more end stations are sending data to the FDR at an aggregate rate greater than 1000 Mbps. Station #1 is transmitting at 600 Mbps and Station #2 is transmitting at 500 Mbps for an aggregate of 1100 Mbps. Incoming packets will begin to accumulate in the input buffers. To ensure that buffers do not overflow, they are equipped with a level indicator. When a buffer's level indicator reaches a preset minimum level (high water-mark), it communicates to the sender via the 802.3x protocol that it must slow down. The level indicator then decreases as the stored frames are transmitted over the bus. When the level indicator crosses a preset minimum level (low water-mark), the port communicates to the sender that it can begin to send frames again.

2.3.2 Forwarding Path

The frame forwarder implements a simple and effective round-robin algorithm to determine which port may transmit onto the bus. The frame forwarder starts at Port 1 and checks to see if a frame is in the buffer. If a frame is available, the port becomes active, one frame is transmitted, and the forwarder moves on to the next port. If a frame is not available at the next port, the forwarder does not activate that port and immediately moves on to the next port. The frame forwarder takes only one frame from a port at a time in order to ensure data from each port are treated fairly. This feature enables the FDR to shape traffic, therefore avoiding "packet clumping," which results in a jittered data stream. This feature is important for delay-sensitive traffic such as video.

The frame-forwarding scheme, combined with per-port buffering, enables the FDR to forward 1000 Mbps of data traffic over its shared bus, even under heavy loads. This performance is significantly better than traditional half-duplex CSMA/CD repeaters, which usually forward only one-third of the maximum bandwidth due to excessive CSMA/CD collisions.

2.3.3 Output

Each output port includes a small buffer, an 802.3z MAC, and an 802.3z PHY. The small output buffer also provides congestion notification. When an input port is determined to be congested, the FDR places the congestion indication frame in this output buffer. Upon receiving this frame, the sender either slows down or stops transmitting until otherwise notified.

To summarize, the key benefits of FDR are:

- massive performance improvement compared to hubs
- simplicity in terms of technology deployment
- full throughput at any packet size
- cost savings versus switching

Today network managers are faced with the challenge of deploying new, bandwidth-intensive applications on their enterprise mission-critical networks. There are now a number of means of solving these challenges. Organizations typically used Ethernet, Token Ring, and maybe FDDI in the backbone five to eight years ago. Now, Fast Ethernet, Gigabit Ethernet, and ATM have emerged as excellent high-bandwidth technologies to solve enterprise bandwidth problems. In addition, a new breed of next-generation switches has emerged to handle these different media, and also provide the quality of service (QoS) features that are necessary to take networks into the next millennium. These switches are known as routing switches. Routing switches are a natural evolution in the inter-networking arena, taking the best elements of layer 2 switching and traditional software-based layer 3 routing and combining them in a hardware-based solution for current and
next-generation networks. A summary discussion of gigabit routers and IP switch alternatives may be found in [9].

2.4 Gigabit Routing Switches for MAN

This section describes the architecture of the gigabit routing switch (GRS), called the PE-4884. The 4884 has 14 vertical slots in the front of the chassis, and 12 of these slots can be populated with individual interface cards (also known as channel cards). The remaining two slots on the front accommodate Enterprise Management Module (EMM) cards. The chassis requires one EMM card to function, and a second EMM card is offered for management, policy, and routing table redundancy. The backplane of the 4884 backplane has a 52 Gbps capacity, ensuring nonblocking wire-speed performance.

2.4.1 Interface/Channel Cards

Different channel cards support connectivity to Ethernet, Fast Ethernet, FDDI, Gigabit Ethernet, SONET, and ATM. These cards provide the physical interfaces that are connected to other routing switches, routers, switches, workstations, servers, hubs, or any network device. The channel card has the following basic functions:

- Sending and receiving data through their physical interfaces
- Performing routing and switching address lookups
- Sending and receiving data from the system packet memory
- Intelligently determining and enforcing layer 4 policies
- Collecting management statistics, SNMP, and RMON data and forwarding the information to the Enterprise Management Modules (EMMs)
- Interpreting or reassigning individual packet priorities

Every channel card is connected to the central memory through two dedicated, nonblocking, full-duplex gigabit channels. In total, there are 26 full-duplex gigabit channels connected in a PE-4884 system into the central memory. Twenty-five of the channels provide the bandwidth for connectivity, and the 26th channel provides system management.

To provide best of class performance in the PE-4884, specialized Application Specific Integrated Circuits (ASICs) were created for the channel cards. For instance, the ASICs on the individual interface modules provide wire-speed intelligent routing data paths for each dedicated channel into memory.

2.4.2 Memory Architecture

The 4884 uses parallel access shared memory architecture to ensure high-speed performance. Although it is very feasible to design a very large Gigabit routing switch based upon cross-bar architecture, this has many limitations. These limitations and the associated impact on networks are described below:

Port-based memory: leads to inefficient use of memory. Memory is divided equally among all ports regardless of their activity or need for memory at a given time. Buffers are quickly used up during a data burst, resulting in unnecessary retransmission of data.

Head-of-line blocking [10]: Packets destined for a busy receive port will often block "head-of-line" traffic that is destined for a non-active receive port because of port-based inbound buffering. Head-of-line blocking creates artificial and unnecessary congestion at the heart of a network.

Difficult to provide reliable QoS support: Cross-bar memory provides static memory pools, which, coupled with head-of-line blocking problems, result in this architecture having limited abilities to deliver prioritized traffic on a port-by-port basis.

The architecture of a shared-memory bus switch was developed to overcome the limitations of cross-bar architectures. This design is very popular within 10 Mbps and 100 Mbps layer 2 switch designers. Using a shared memory bus, all ports access a memory pool placed locally on a switching module or workgroup switch. A port can access the central memory through a common bus when an arbitration device grants it access. For Ethernet and smaller Fast Ethernet switch designs, this architecture eliminated the port-based memory and head-of-line blocking problems associated with cross-bar switches. It would seem, therefore, that a shared memory bus architecture would be an excellent choice for a Gigabit Ethernet routing switch. However, it was found that it is currently impossible to run a bus arbitration scheme fast enough to provide nonblocking performance for a high-capacity gigabit routing or switching fabric.

To avoid the blocking problems associated with shared memory bus architectures when used with Gigabit Ethernet, a parallel access shared memory architecture that delivers wire-speed performance regardless of media was designed. This architecture provides a foundation for frame-based QoS solutions and the wire-speed delivery of advanced network services such as IP multicasting.

Parallel access shared memory is somewhat similar to the shared memory architecture designs, where all ports share a central memory location. However, unlike shared memory architectures, parallel access shared memory provides every port on every module with a dedicated path into and out of the central memory fabric located at the back of the switch. A centralized memory pool makes it possible for all ports to have simultaneous full-duplex gigabit access to the central memory at any given time. Complete accessibility, combined with the available 52 Gbps of available backplane capacity, provides true nonblocking performance. This system also eliminates the need to replicate and store identical multicast traffic on multiple egress ports. The central storage of traffic provides the ability to pull outbound traffic from dedicated port-based priority queues (eight per port). To ensure fair treatment of priority, a sophisticated weighted fair queuing algorithm is used on each port and conforms to IEEE 802.1p. This allows end users to take full advantage of the eight levels of priority that can be defined through IP precedence bits.

107
direct frames. The OMF can be a type of traffic (for instance, HTTP), traffic from a specific application (Point Cast), or even traffic from a restricted site on the network.

3. Supporting Distributed Multimedia Applications

One of the many potential distributed applications that can benefit from the gigabit bandwidth is the multimedia application. Distributed multimedia applications can be classified as video conferencing for live video stream and on-demand video delivery for video titles. As video conferencing is still a CPU-bound application, the typical performance bottleneck is thus in the CPU speed instead of communication bandwidth. We will focus on the on-demand video titles instead because they are an I/O intensive application.

The main challenges of retrieving on-demand video titles are (1) large data size: although uncompressed video titles can be significantly compressed, the size of a high-quality compressed video title is still in the range of 3-5 Gigabytes; (2) real-time constraint: a video frame needs to be displayed at the receiving host in time such that the compressed frame can be decompressed and displayed without jitters; and (3) supporting concurrent accesses: in order to amortize the system cost, a distributed on-demand video service should support as many concurrent accesses as possible. Considering all these requirements together, the system design is challenging.

For example, in the past few years we have designed and implemented a controllable software architecture for an on-demand video server. We studied several disk striping schemes in the storage subsystem and examined the impact of these schemes on the utilization of system resources. In the network subsystem, we adopted a server-driven approach for investigating the MPEG-2 video delivery. The prototype server is based on an SCI shared-memory multiprocessor with a mass storage system consisting of multiple disk arrays. Using 30 RAID-3 disk arrays, preliminary experimental results show that the prototype server can potentially support more than 360 high-quality video streams with careful design and coordination of the different system components [3]. The amount of traffic generated from the server was 2.880 Gbps (i.e., 360 * 8 Mbps).

Because a typical distributed on-demand video service requires support beyond gigabit bandwidth, it is our belief that many challenges lie in the design of FDR and gigabit switch. Many of the designs under our considerations are actually multimedia-system-friendly. For typical video conferencing and on-demand video titles, in terms of the connection set-up, we are considering:

1. Implementation of the RSVP scheme for signalling the new connections: our vision is that an application can renegotiate if RSVP fails to find resources;
2. On the performance side, new emerging standards such as IEEE 802.1p and 802.1Q. 802.1p deals with the

\footnote{A typical video conferencing session uses up to 30% CPU time even with the assisting hardware capture board [11]. Software compression/decompression will use virtually 100% of the CPU for the real-time session.}
specifications of mechanisms in the MAC to expedite delivery of time-critical traffic and to limit the extent of high-bandwidth multicast traffic (e.g., MBONE). Priority schemes are intended to determine which traffic is most time critical. In order to further support priority-based real-time communications, we also implement eight priority queues on each port of the FDR and the 4884 switch. The overall goal of 802.1Q is to develop an architecture and protocols for the logical partitioning of a LAN that provides separate instances of the MAC service to administratively defined groups of users. This scheme potentially can help partition the applications into different groups (e.g., multimedia vs. conventional application) such that different MAC service can be provided.

Note that these two schemes are still expected to be finalized over the next few years. Current drafts (draft version 8 for 802.1p and draft version 7 for 802.1Q) can be referred to [12, 13].

3. We also envision further integrated solutions with policy-based QOS. The levels of implemented policies will include port level at the GNIC, FDR, and 4884 switch; box level within a FDR and 4884 Switch; and end-to-end level between the GNIC, FDR, and 4884 switch.

We believe that these schemes will benefit the integration of video conferencing and on-demand video service with the conventional applications. For instance, an early experiment (in addition to the gigabit network interface performance that we report later in this article) indicated that the design of round-robin forwarding in the FDR can support high performance in transmitting concurrent video data. Recently, we have been working with Microsoft to experimentally demonstrate that a MPEG-2 video stream will not be affected by the background traffics by using the round-robin forwarding in an FDR.

3.1 Experimental Results

There were two major goals of our performance evaluation experiments: first, to test how a gigabit LAN performed in the basic metrics of throughput for application level messages; and second, to examine how concurrent video delivery is supported over a gigabit LAN. These experiments were designed to determine: the bottlenecks in the current systems and to identify the bounds on performance; the peak and sustained throughput of application messages, which determine how well a gigabit Ethernet LAN would support an I/O-intensive application; and the jitters in the delivery of concurrent video data for on-demand video services.

We focused more on the end-to-end performance between the client and the server nodes. End-to-end performance is especially important in a gigabit Ethernet LAN where the link bandwidth and the switch throughput may not be the bottlenecks, but most likely the interaction between the network interface and the host operating system will be. In fact, we find the limits to the performance of the gigabit Ethernet are set by the host platform. The host operating system (e.g., Linux and NT) and the system bus (e.g., PCI bus on PC) are the primary bottlenecks.

We note that although we are measuring end-to-end performance, the relative novelty of gigabit networks meant there are few performance-monitoring tools for the network or in the hosts. In our case, we require performance monitoring at the adapter interface, where currently only limited benchmarking utilities (e.g., netperf and netpipe) are available. Our own measurements thus were limited to inserting timer probes close to the application. Clearly, measurement tools that allow better access to timing measurements at IP and below are required in the future for higher fidelity measurements.

We first measure the throughput on the gigabit Ethernet LAN between a pair of nodes. We provide comparisons of the netperf and netpipe throughput with our own benchmarking method. We then benchmark the jitter across the network for supporting concurrent on-demand video streams.

3.1.1 The Experimental Setup

Our testbed consists of one PentiumPro and one Pentium II, one working at 233MHz and the other at 300 MHz, where both run either Linux or NT. The G-NIC PCI card fits onto a 32-bit slot on both machines. The device driver used for Linux was obtained from [14]. The testbed is shown in Fig. 5.

![Figure 5. Testbed configuration.](image)

With the hardware and operating systems in place, we were able to measure the experimental performance of typical data communication applications. The netperf is designed around the basic client-server model and is available from the HP labs [4]. There are two executable processes, neserver and netperf, which are used for benchmarking. The Netserver program needs to run on the remote system, which will be contacted by netperf. The netperf code uses a control connection to send the parameters of the test and the results to and from the remote connection. The control connection is always a TCP connection using BSD sockets. A separate connection is opened for the measurement, which uses the protocols appropriate for the test. Various parameters like packet size, socket buffer size, and the like, can be specified by the user. Tests can be run over varying periods of time. Note that Netserver only replies with a short acknowledge message to the client after receiving the variable-size packets. This behaviour thus produces a nonsymmetric communication pattern whose results tend
to be optimistic. Therefore, we treated the results from netperf tool as the peak performance.

On the other hand, we also like to gain the average performance in a peer-to-peer symmetric communication pattern. The experiment involved a round-trip transfer of a text message of a given size from the client to the server and back (Fig. 6). In other words, the server was implemented as an echo-server. The size of the messages was varied in discrete steps from 128 bytes to 1 Mbyte. Round-trip delays were measured for each experiment at the client, and throughput values were computed from the measured delay. (The measured delay was the total elapsed time for one round-trip transfer of an application message.) Lost cells and retransmitted cells did not change the amount of data in the measurements. They were accounted for in terms of the additional delay incurred at the application level for the delivery of the complete message.

Each experiment was run several times in order to minimize the randomness inherent in such tests. The process loads at each workstation were kept small, with only the bare minimum of processes necessary to run the kernel and with the Window interface enabled. The application was instrumented with get_time_of_day() timer probes. Separate time measurements were made to eliminate the time overhead introduced by the execution of the probe calls. This ensured more accurate recording of the delay measurements. Timing calculations are done for many such trials, and only the values within a 95% confidence interval are used; the outliers are rejected in calculating the average throughput. The program also sets the TCP_NODELAY option, which can turn off the Nagle’s algorithm so that packets are sent as they come and are not bunched into larger TCP packets.

### 3.1.2 Benchmarking Conventional Data Communication

In this section we look at the performance results obtained with the G-NIC cards on our testbed. We initially look at the performance obtained by using the cards on Linux and NT platforms with benchmarking codes like netperf and our own benchmarking code.

#### Peak throughput using netperf

Fig. 7 depicts the peak performance using the netperf tool on the Linux and NT platforms. Note that netpipe was also used for benchmarking, and because it reported results close to the netperf experiments it is not discussed further.

![Pseudo code of the tool for measuring average performance.](image)

Figure 6. Pseudo code of the tool for measuring average performance.
Figure 7. Peak throughput performance using netperf measurement tool.

The message sizes were varied from 2048 bytes to 4M bytes (labeled as "1" to "12" as they map to 2^1 and 2^12 KBytes). The peak throughput obtained in Mbps is given in Fig. 7. The socket buffer sizes in the sender and receiver sides were set to 64K, Linux by default uses a maximum TCP window size of 32K. The experiments with netperf on the Linux platform indicate a performance trend that for smaller packet sizes, the system obtains a good throughput that is steadily increasing. After the 2^4 = 16 KByte packet size is reached, there is a slight drop in performance, and again an increase to give us an almost steady throughput close to 190 Mbps from thereon. The reason for this could be that with larger data packets, TCP sends the data as it arrives. With the really small ones, TCP is capable of grouping packets together to gain efficiency and so we get good results. It is interesting to observe that the netperf over Linux platform reaches the peak performance close to 190 Mbps quickly and stably.

However, the NT platform over the same PC machines did not achieve the same level of performance. We were able only to reach the peak performance of a little over 90 Mbps on the NT platform over a PC. In order to identify the potential bottlenecks for causing this low efficiency, we did the same NT experiments based on a DEC machine with Alpha CPU of 533 Mhz. We observed that the peak performance improved to reach the equal level of 180 Mbps when the message size was 2^4 = 32 KByte. These experiments indicated that the NT device driver for the GNIC card still sufficiently performs the communications. However, the design of NT operating system is the key factor in determining the underestimated results, especially when the CPU’s speed is not high. The exact reasons are beyond the scope of this article, and we will simply point out that this issue needs further investigation. On related experiments, Sun’s Solaris 2.5.1 operating system achieved speeds of around 488 Mbps running on an UltraSPARC.

We repeated the experiments with the window size changed to 64K. We noticed that with packet sizes greater than 64K we were able to reach a peak throughput of about 205 Mbps. Therefore, the peak performance over PCI-based Pentium 233 MHz PC is about 180–200 Mbps in the application level. It is our belief that the PCI interaction and the internal protocol stack within the operating system are key factors that lead to this level of performance results. As raw device testing in the hardware level has shown that the G-NIC cards can perform at 700–800 Mbps at the hardware/firmware level, we ruled out the possibility of hardware/firmware defects.

This observation prompted us to look at the various points on the protocol stack that have resulted in the drop of performance from 700–800 Mbps to about 190–200 Mbps at the application level. We could take a look at the device driver level (i.e., yellowfin.c) in the implementation. However, the experiment design was not trivial. Our goal is to measure the performance of device driver level by modifying the device driver code to see how fast it is able to transmit data. The yellowfin driver [14] uses two ring buffers: an Rx ring and a Tx ring. A full frame size buffer (skbuff) is allocated for the Rx and Tx ring. If the incoming frame is smaller than a cut-off value, a new buffer is created and the data are copied onto this buffer. If the frame is larger than the cut-off value, it is sent up the protocol stack. The driver has two independent threads: Send packet thread, which takes care of the Tx ring; and Interrupt handler, which takes control over Rx ring.

Changes were made by introducing a timer probe (do_get_time_of_day command) before and after the transmission code. The results of this experiment had shown that a PCI-based machine was able to achieve a peak performance between 300 to 350 Mbps in the device driver level. We are also planning to look at the performances at the transport (i.e., TCP/UDP) and network layers (i.e., IP) in the future. Nevertheless, the performance of device driver study indicated that the PCI interaction is indeed one key factor.

Average Throughput. Though the netperf successfully measured the peak performance at the application level, many applications may not always impose the same traffic patterns as netperf does. For example, in order to measure the peak performance, netperf continuously sends out the packet into the TCP’s windows (via socket interface) with the best possible grouping. The return packet size by the server is always small (e.g., a couple of bytes) to speed-up acknowledgment speed. This kind of communication is good for measuring the peak performance but does not indicate a peer-to-peer symmetric traffic (i.e., average throughput).

Our benchmarking code, as explained earlier in Fig. 6, was used to find the average throughput on the Linux platform. The results of our experiment are shown in Fig. 8. Because we used the same measurement tool for our previous study on ATM LANs in [11], we depict both of the performance trends in the same figure. The ATM LAN was constructed over Sun’s SPARC-2 machines each with a 155 Mbps OC-3 link to the Fore’s ASX-100 switch. Notice that gigabit Ethernet had a much faster rising slope on sustaining the average throughput to 160 Mbps, whereas the ATM saturated at 48 Mbps using AAL-5 packets.

4 To the best of our knowledge, there is no existing measurement tool available in this level on gigabit Ethernets.
results from this Netbench test are summarized in Table 2. Note that the maximum server throughput of 157 Mbps was observed with three clients. We are in the process of setting up a similar Linux and Solaris-based platform. This will enable us to conduct more rigorous testing and application development using the FDR.

**Supporting High-Quality Streaming Videos.** The performance of the G-NIC cards was further evaluated with an experiment on concurrent video transfers. The basis of this experiment is to find out how many simultaneous video transfers can be supported on the network while maintaining a certain quality of service. We believe the multimedia-based communications (e.g., on-demand video titles and live video conferencing) will be a major application that gigabit Ethernet needs to support. Therefore, one important measure of performance will be to find out how many simultaneous video streams can be supported with guaranteed quality. As the GNIC is a fundamental component for supporting this service, we first measure the video performance supported by a pair of GNICs in our testbed.

Designing experiments to measure concurrent accesses for on-demand video titles is difficult. First, a reasonable buffering scheme needs to be adopted. In order to guarantee acceptable application performance, we implemented the two-buffer scheme [15] in our experiments. One buffer is used for the retrieval of the video frames from the storage subsystem and the other buffer for the transmission of video frames to the remote clients. The client also uses one buffer for the network and another buffer for display. Assume there are $x$ video frames for a 30-frame-per-second video in the buffer that is being transmitted by the network subsystem. In order to provide continuous playback, the entire buffer must be delivered and subsequent video frames in the other buffer must be ready for transmission within $x$ frame intervals ($x \times 33.33\text{ms}$). The use of these two buffers is swapped after $x$ frame intervals. There are two dedicated buffers assigned to serve each video stream.

Next, we need to define the performance metrics for the measurement. In our experiments, we use the following quality of service (QoS) parameters at the application level to determine the maximum number of concurrent accesses that can be supported by the network. We defined the (application-level) jitters as the number of miss-deadline retrievals. It reflects the total number of network transmissions that missed the deadline. The deadline depends on how many video frames were retrieved for each remote read request.

The basic idea of the experiment is that the server gets data from a storage device into its buffer. This buffer fills up the network buffer and the data are sent over the network to the network buffer on the client side. The display buffer on the client side picks up data from the network buffer and sends that data to the display. For jitter not to be experienced during display, the network buffer on the client side must always have data to send to the display buffer. This is possible if the data come in fast enough from the server side to the client side over the network. For each experiment, we measured the performance for each

---

**NT Server Throughput with the NetBench Tool**

<table>
<thead>
<tr>
<th>Number of Clients</th>
<th>Server Throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111.3</td>
</tr>
<tr>
<td>2</td>
<td>146.7</td>
</tr>
<tr>
<td>3</td>
<td>156.6</td>
</tr>
<tr>
<td>4</td>
<td>152.2</td>
</tr>
<tr>
<td>5</td>
<td>148.0</td>
</tr>
<tr>
<td>6</td>
<td>145.2</td>
</tr>
</tbody>
</table>
Table 3
Number of Concurrent Accesses Supported for the Gigabit GNICS

<table>
<thead>
<tr>
<th>Display Speed (frames/sec)</th>
<th>Rate</th>
<th>Frame Size</th>
<th>Requested Block Size (# frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 (NTSC)</td>
<td>4 Mbps</td>
<td>16 KBytes</td>
<td>32 32 32 32 32 N/A</td>
</tr>
<tr>
<td></td>
<td>8 Mbps</td>
<td>32 KBytes</td>
<td>16 16 17 17 18 19</td>
</tr>
<tr>
<td></td>
<td>16 Mbps</td>
<td>64 KBytes</td>
<td>6 7 8 9 9 9</td>
</tr>
<tr>
<td></td>
<td>32 Mbps</td>
<td>128 KBytes</td>
<td>2 3 3 3 3 3</td>
</tr>
</tbody>
</table>

Table 4
Number of Concurrent Accesses that Can Be Supported Using a Single 16 GByte Disk Array Source. From [3].

<table>
<thead>
<tr>
<th>Display Speed (frames/sec)</th>
<th>Rate</th>
<th>Frame Size</th>
<th>Requested Block Size (# frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 (NTSC)</td>
<td>4 Mbps</td>
<td>16 KBytes</td>
<td>4 8 16 23 29 33</td>
</tr>
<tr>
<td></td>
<td>8 Mbps</td>
<td>32 KBytes</td>
<td>4 7 11 14 15 17</td>
</tr>
<tr>
<td></td>
<td>16 Mbps</td>
<td>64 KBytes</td>
<td>3 5 6 8 8 NA</td>
</tr>
<tr>
<td></td>
<td>32 Mbps</td>
<td>128 KBytes</td>
<td>2 3 3 4 NA NA</td>
</tr>
</tbody>
</table>

active retrieval process at the client site.

The experiments were repeated by incrementing the number of active processes until the system (including server and network) could no longer provide an acceptable quality of service (i.e., the average jitter ratio was greater than 1%). In this way, we could evaluate how many concurrent accesses could be provided with guaranteed quality at the client side.

Table 3 illustrates the network performance on supporting concurrent video accesses. The range specifications of MPEG-2 between 4 and 32 Mbps are emulated in these experiments. The numbers that are listed in Table 3 are the video streams that experienced jitters of less than 1%. Due to bus contention among the CPU, storage, memory, and network interfaces, we have observed that a sustained 128 Mbps was achieved for supporting 32 streams of 4 Mbps videos. The performance was bounded by the severe contention on the system resources. Increasing the video block sizes seems to improve the numbers of supported video streams for 8- and 16-Mbps videos. The sustained throughput was \(8 \times 19 = 152\) Mbps and \(9 \times 16 = 144\) Mbps for the 8 and 16 Mbps video streams. When the video quality is extended to very high end, such as 32 Mbps, the system only supports up to \(3 \times 32 = 96\) Mbps as the sustained network throughput.

However, we should point out that in a typical video server configuration, the sustained network throughput may not be the only performance bottlenecks. Table 4 lists the performance of a storage system by connecting a 16 GByte disk array to the video server via a fast-and-wide SCSI-2 bus with 20 MByte-per-second channel bandwidth.

The storage performance indicates that the number of streams that can be supported by a high-performance disk array is far less than the supported numbers on the network system. It is now clear that a proper buffer size such as 64 frames should be adopted in order the balance the storage and network systems. Further design issues concerning the storage system, such as striping across a number of disk arrays [3], can be adopted to reduce the buffering requirement, thus utilizing the network system more efficiently.

4. Conclusions

As mentioned earlier, the need for gigabit bandwidth can be attributed to a fast-growing set of applications, such as streaming video, virtual reality, distributed computing, and so on. Simultaneously, end systems are gaining capability in terms of being able to support such bandwidths at the bus level. At present, typical Pentium-based desktops have maximum bus bandwidth of around 250 Mbps. High-end SMP PC servers and workstations have bus bandwidths of around 600 to 800 Mbps. It is expected that future desktops and servers will have even higher bus bandwidths. These systems can potentially exploit the gigabit bandwidth provided by the systems described in this article.

From the performance results, we observe that gigabit speeds at the interface card level can be achieved. However, other layers in the network protocol stack such as transport (TCP) and network (IP) layers need to be enhanced to offer the application such high bandwidths. High-end workstations using SUN UltraSPARC and DEC Alpha chips are able to achieve around 400 to 500 Mbps bandwidths even at the TCP level. In the near future, processing latency can be further reduced through cleverer implementations, protocol off-loading, and simplified transport protocols.
We are currently examining high-performance modifications to TCP [16] that will, among other aims, increase the TCP window size from 64Kbytes to up to 1 Mbytes. These extensions are designed to improve TCP throughput by allowing more outstanding unacknowledged data. However, as processing latency is a very important concern in high-speed networks, we are also examining other light-weight transport protocols that will reduce processing latency overhead.

To summarize, this work presented the design and architecture of a gigabit interface card, a full duplex gigabit hub, and a gigabit routing switch. An experimental testbed was set up at Washington State University and University of Florida to test the performance of the network up to the transport and application layers. The results indicate that at the transport layer, approximately 190–200 Mbps can be achieved on lower-end Pentium workstations and 400–500 Mbps can be achieved on high-end SUN UltraSPARC and DEC Alpha workstations.

As workstations get more powerful and bus bandwidths increase, the observed throughput will be higher. Given the rapid advances in CPU and system buses, the gigabit personal communication is more feasible than ever. It is worth noting that even when the gigabit network bandwidth is achieved, the electronic processing and bus bandwidths will still peak at a few Gbps. This is in contrast to the potential 25,000 Gbps available on the optic fiber, and we are currently using only 0.004% of this bandwidth! To fully utilize the bandwidth, newer technologies and faster processing architectures need to emerge.

Acknowledgements

The authors are grateful to Mr. Donald Becker at NASA’s Center of Excellence in Space Data and Information Sciences, Goddard Space Flight Center, for assistance with the yellowfin.c device driver. This work is supported in part by Packet Engines, WTC RTD Grant 98-C3, and NSF under CCR-0096112 and CCR-9874726.

References


Biographies

Bernard Daines is the founder, president, and chief executive officer of World Wide Packets. He has over 30 years’ experience delivering innovative solutions in the high-speed networking industry, and is widely recognized as an expert in Ethernet technology. Daines founded World Wide Packets in 1999 to develop the first cost-effective and future-proof solution for optical broadband connectivity. In 1994 he founded Packet Engines, a gigabit Ethernet routing-switch manufacturer that was sold to Paris-based Alcatel in 1998 for $325 million. In 1992 Daines co-founded Grand Junction Networks, a fast Ethernet switch manufacturer that was sold to Cisco Systems in 1995 for $350 million. Daines has been instrumental in the development of the IEEE standards and innovative solutions for fast Ethernet and gigabit Ethernet. His contributions to the advancement of the networking industry have been widely recognized world wide, and he is often called upon to provide his insightful perspective at conferences and industry forums. Daines holds a B.Sc. in computer science from Brigham Young University.

Jonathan C.L. Liu received the Ph.D. degree from the Department of Computer Science and Engineering at the University of Minnesota in 1996. During 1996-1999 Dr. Liu was Assistant Professor of Computer Science with the School of Electrical Engineering and Computer Science at Washington State University. Since August 1999 he has been with the CISE Department at the University of Florida, where he is now Assistant Professor. His current research interests include high-speed networks, multimedia communications and databases, parallel processing, and artificial intelligence. Dr. Liu is the recipient of the National Science Foundation CAREER Award and is a member of IEEE and ACM.