A REAL-TIME MESSAGING SYSTEM FOR TOKEN RING NETWORKS

Alfred C. Weaver

M. Alex Colvin

Computer Science Report No. TR-87-11
March 1987

Submitted to *Software -- Practice and Experience*, March 1987.

A REAL-TIME MESSAGING SYSTEM FOR TOKEN RING NETWORKS

Alfred C. Weaver and M. Alex Colvin
Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, Virginia 22903

ABSTRACT

The Computer Networks Laboratory at the University of Virginia has developed a real-time messaging service which runs on IBM PCs, PC/ATs, and compatibles when interconnected with a Proteon ProNET-10 token ring local area network. The system is a prototype for a real-time communications network to be used aboard aircraft or ships. The system conforms to the IEEE 802.2 Logical Link Control standard for type I (connectionless, or datagram) service, with an option for acknowledged datagrams.

The application environment required substantial network throughput and bounded message delay. Thus the development philosophy was to initially emphasize performance and to offer only primitive user services. After providing and measuring the performance of a basic datagram service, the intent is to add additional user services one at a time and to retain only those which the user can "afford" in terms of their impact on throughput, delay, and CPU utilization.

The current system is programmed in *C*. The user interface is a set of *C* procedure calls which initialize tables, reserve buffer space, send and receive messages, and report network status. The system is now operational and initial performance measurements are complete. Using this system, an individual PC can transmit or receive approximately 200 short (about 100 byte) messages per second, while the PC/AT operates at nearly 500 short messages per second.

I. EXPERIMENTAL ENVIRONMENT

The network is small, consisting of five PCs and one PC/AT as shown in Figure 1. Each PC is used as a load generating station; the PC/AT is used as a network monitor. Each PC has a Proteon ProNET-10 [PROT85] interface board set for an individual hardware-selected address; the PC/AT uses a network monitor board which allows it to capture all network traffic regardless of destination. Each PC is connected via a 3-meter shielded twisted pair cable to a port on a wiring center; wiring centers are interconnected via 100-meter shielded twisted pair cables.

Figure 2 shows the mapping of the ISO OSI model to this environment. The Medium Access Control (MAC) sublayer of the Data Link Layer is contained in hardware and firmware on the ProNET-10 interface board. The Logical Link Control (LLC) sublayer is implemented in the programming language *C* and resides within the user's program as a part of the run-time system.

Using the conventions of <u>request</u>, <u>confirm</u>, and <u>indication</u> as suggested by the IEEE 802.2 Logical Link Control standard [IEEE85], the LLC supports a primitive datagram service and, optionally, acknowledged datagrams. These services and their timing definitions are shown diagrammatically in Figure 3. The implementation currently conforms to LLC type I (connectionless) with an option for packet acknowledgements. Type II messages (connection-oriented) are not currently supported.

II. PROGRAMMER INTERFACE

The programmer uses the LLC interface by linking into his *C* program as many of the following communications primitives as he needs. Examples are shown in *C* code style for clarity to the programmer. Communication occurs through *sockets*, which are equivalent to LSAPs (IEEE 802.2 Link Service Access Points). A network address consists of two octets; one is the socket number, the other is the hardware address of the token ring interface.

```
LLCon (socks)
int socks;              /* socket number */
```

Initializes the network interface. Table space for *socks* number of sockets is allocated.

```
LLCoff ( )
```

Disables and closes the socket interface.

```
LLCopen (sock)
int sock;               /* socket number */
```

Opens the socket numbered *sock*. A socket must be opened before use. Sockets must be even-numbered integers in the range 2 <= *sock* <= 254.

```
LLCclose (sock)
int sock;               /* socket number */
```

Closes the socket numbered *sock*. The socket must be idle (no pending transmit or receive calls).

```
LLCoption (sock,xindication,rindication,xticks,rticks,priority,crc,retry)
int sock;               /* socket number */
int (*xindication)(), (*rindication)(); /* transmit and receive functions */
int xticks, rticks;     /* timeout counts */
int priority;           /* priority class */
int crc;                /* flag to force computation of CRC */
int retry;              /* number of automatic transmission attempts */
```

For socket number *sock*. *xindication* and *rindication* are pointers to functions to be called when a packet has been transmitted or received, respectively. Variables *xticks* and *rticks* are the count of the number of timer interrupts allowed for the operation to complete (a zero value never times out). The value of *priority* sets the transmission priority of the message, in the range 0 (lowest) to 7 (highest). *crc* is a flag which, if set, requires that the packet be explicitly acknowledged by a reply message. When the token ring's frame acknowledgement bit shows that the destination did not receive a packet correctly, *retry* tells how many times the transmitter should transparently send the packet before declaring an error.

```
LLCxmit (sock, nap, ip, isize)
int sock;               /* socket number */
char *nap;              /* pointer to network address */
char *ip;               /* pointer to packet information */
int isize;              /* size of packet */
```

Delivers to the MAC engine the packet pointed to by *ip*, of length *isize*, to socket number *sock* for delivery to the network entity who address is pointed to by *nap*.

```
LLCrecv (sock, nap, ip, isize)
int sock;               /* socket number */
char *nap;              /* pointer to network address */
char *ip;               /* pointer to packet information */
int isize;              /* size of packet */
```

Enables reception of a packet at socket *sock.* The programmer provides *nap.* a pointer to a 16-bit buffer for the packet's source address, *ip.* a pointer to the buffer which will hold the packet, and *isize.* an integer for describing the length of the received packet.

Note: **LLCxmit** and **LLCrecv** move messages between the LLC entity and the MAC engine in the same computer (not end-to-end). This frees the CPU to operate in parallel with the network hardware. Using 802.2 terminology, the procedure call represents the data *request.* the return from the procedure represents the *confirm.* and an appropriate change in the status byte provided by **LLCstatus** represents the *indication.*

```
LLCreset (sock, ch)
int sock;               /* socket number */
char ch;                /* character "x" or "r" */
```

A socket is bidirectional and so can send and receive simultaneously. This operation resets any pending operation on the transmit ("x") or receive ("r") side and releases the associated buffer.

```
LLCstatus (sock, ch, sp);
int sock;                /* socket number */
char ch;                 /* "x" or "r" */
int *sp;                 /* pointer to status */
```

The status of a socket is obtained by calling LLCstatus. The returned value indicates operation pending, no operation pending, I/O in progress, operation failed, or operation timed out.

Finally, every operation on a socket returns an operation status: operation accepted, invalid socket, duplicate socket, too many sockets, socket busy, packet too large, or transmit/receive argument neither "x" nor "r".

III. PERFORMANCE RESULTS

A. Terminology

The following figures show performance results measured at various layers. The PHY (physical) layer represents the basic transmission process accomplished in hardware. The PHY transmit measurements are from the point of view of a packet which has already been enqueued in the ProNET-10's packet buffer; these measurements are interesting because they are an absolute lower bound on all other delay measurements. The MAC (medium access controller) layer represents performance from the point of view of a packet which has just emerged from, or is just ready for, LLC services. For transmission, this means the packet has already been fully framed by the LLC operation in the host and is now ready for transmission across the PC's bus to the Proteon packet buffer. For reception, this means that the Proteon board has received a packet and finished MAC-level operations (for example, computing a hardware error check) and the packet is now ready for transmission (again across the PC's bus) to the host's main memory. The LLC (logical link control) layer represents performance at the level of the programmer's *C*-language interface; although it is the highest layer of these three, it is the lowest layer whose performance is directly observable by the application process.

Using OSI terminology, the application process in the host (the user's *C* program running on the PC) interfaces directly with the LLC (our datagram communications service, written in *C* and linked into the user's program), since OSI layers three through seven are absent. For any packet, that portion of its delay attributable to PHY is just the packet length in bits divided by the transmission speed of 10 Mbps. Transmit time at MAC is the inverse of the maximum frequency at which a single packet, enqueued in the host's main memory, could be repeatedly sent to the ProNET-10 hardware. Transmit time at LLC is the inverse of the maximum frequency at which packets can be delivered to MAC by the **LLCxmit** primitive.

B. MAC and LLC Operations

Figure 4 shows the timing for the primitive MAC operations of buffer read, buffer write, and buffer transmit when using a Leading Edge model "D" (IBM PC-compatible) with an Intel 8088 CPU and 6.17 MHz clock. The curve labelled "buffer read time" shows the time required for the host (the PC) to read a frame from the packet buffer into the host's main memory utilizing direct memory access (DMA). The curve labelled "buffer write time" shows the timing for writing a frame from host memory into the hardware packet buffer, again using DMA. The curve labelled "buffer transmit time" shows the time to transmit a packet through the PHY interface.

The fourth curve shows the end-to-end delay of a packet and is the sum of the previous three. For any packet, the end-to-end delay is the sum of the buffer write time in the transmitter, the buffer transmission time after gaining network access, and the buffer read time in the receiver. Thus this curve gives an accurate picture of the time elapsed from the moment the packet is enqueued in the transmitter's main memory until it is completely delivered to the receiver's main memory.

Figure 5 presents similar results measured at the LLC. It shows the transmit, receive, and end-to-end times between two peer LLC layers. Figure 5 illustrates the cost (i.e., delay) of the minimum possible LLC service, namely packet framing.

C. Comparison of Layers

Figure 6 provides additional perspective on packet-level performance at each layer. The PHY curve represents the hardware transmission time and is an absolute lower bound on packet transmission time. The MAC curve shows the time required at the level of the PC-to-network interface. The LLC curve shows the performance observed by the application process if it measured the **LLCxmit** operation. Note that all of these measurements are transmit (not end-to-end) times.

An interesting observation is that MAC-layer services require a minimum of about a millisecond, regardless of packet size, due primarily to the overhead of starting up the DMA channel to transfer the frame from main memory to the packet buffer. As packet size grows beyond 100 bytes, the amount of information transferred begins to influence the DMA timing. A MAC-layer transmit operation occupies about 1 ms for a short (less than 100 byte) packet, growing to 7.5 ms for the maximum length 2,048-byte packet. Similarly, the LLC service requires a minimum of 4 milliseconds to accomplish its task, independent of packet length, increasing to 11 ms for the maximum length packet.

D. LLC Optional Services

Figure 7 illustrates the "cost" of some optional LLC services. The LLC transmit time curve from Figure 6 is repeated for reference. The curve labelled "LLC transmit + ACK time" shows the time required for an acknowledged datagram, i.e., a single packet which requires a packet acknowledgement. As expected, requiring a return packet to acknowledge the outgoing packet essentially doubles the delay.

The curve labelled "LLC transmit + CRC time" shows the delay encountered when using a software CRC algorithm (computed byte-by-byte) to assure end-to-end validity of the message. It is often overlooked that the frame check sequence (usually a cyclic redundancy code or checksum) in all MAC-layer protocols provides error detection between MAC engines, i.e., between network interface boards in different computers. But because the frame check sequence (FCS) is added after the packet is delivered to the MAC engine, and stripped off in the receiver when the packet has been received, the FCS can provide no protection against errors which might occur in the host while the message is being transferred between MAC and LLC (in this case, between the main memory and the packet buffer across the PC's bus).

While these transfers are error-checked in minicomputers, there is often no such checking with personal computers. To achieve reliable end-to-end (i.e., main memory to main memory) error detection, some form of error checking (e.g., CRCs or checksums) must be used on the message itself at the LLC (and possibly higher) layer. This curve shows that, as expected, the overhead of CRC computation is linear with the message length, and for large messages it dominates the transmit delay.

The fourth curve in Figure 7, "LLC transmit + ACK + CRC time", provides timing information when both options are selected. This curve is significant because it shows how rapidly delays accumulate even for simple services. Even a short (100 byte) packet requires 16 ms when both acknowledgements and CRCs are utilized.

E. Comparison of PC vs. PC/AT

Figure 8 presents a comparison of the PC vs. the PC/AT (a Compaq Portable 286). Figure 8(a) compares MAC-layer transmit times; Figure 8(b) compares LLC-layer transmit times. Note that the PC/AT is faster for short messages, but slower for long messages. This reveals a difference in the DMA channel on the two machines. The PC/AT has a faster DMA access (startup) time than the PC, but a slower per-byte transfer time. For MAC services (which are all DMA-bound), the processing speed of the PC/AT is not a factor; the PC is faster for packets longer than 300 bytes. For LLC operations (which are more

CPU-intensive), the PC/AT demonstrates superior performance for packets shorter than 1,400 bytes.

IV. NETWORK PERFORMANCE MONITOR

We have also developed a real-time network monitor which presents color bar charts of recent network utilization. The screen display is shown in Figure 9. The upper third of the screen displays the current time (since network initialization) and provides space to notify the network operator of exceptional events. The middle third of the screen displays a moving record of recent network activity. Utilizing a user-selected sampling interval and vertical scale, the monitor draws a vertical line depicting the network traffic in units of packets per second. In the lower third of the screen the same traffic is displayed in units of bits per second. Within each display area, an alphanumeric line shows the current rate, average rate, and peak rate (since last initialization) of traffic observed. The "title line" also shows the units of measurements (packets/sec or bits/sec) and the user-selected vertical scale.

Once each sampling interval, the performance of the network is updated internally. Data gathering always has priority over data display. When processing time permits, the display is shifted left and the most recent network measurements are displayed in the rightmost pixel positions. Using the 500 ms default sampling interval, the screen provides a visual history of network activity for the previous 3 minutes.

Using a sampling interval of 10 ms, the display shows network activity for the previous 3 seconds.

The network monitor can also be configured to display traffic from a specific source address and/or traffic to a particular destination address. "Match-all" mode may be selected for either source or destination as well, and when used for both results in the total network traffic picture as described above.

CONCLUSIONS

We conclude from our work with the prototype that it is possible to build an effective, although primitive, data communications service using simple hardware and software. We can establish a real-time (i.e., delays in the milliseconds) datagram service, providing LLC type I service, by providing about a dozen LLC primitives.

In combination with the Proteon ProNET-10 token ring and our *C* language user interface, the PC supports up to 200 short (about 100 byte) messages per second while the PC/AT supports nearly 500 short messages per second. The network monitor displays two real-time, color bar charts of network utilization (in packets/sec and bits/sec) which provide a visual history of network activity. The real-time monitor has proven extremely useful for network debugging.

As we add additional services (e.g., virtual circuits, acknowledgements, end-to-end CRC codes, file transfer, program upload/download, etc.), we will do so incrementally so that the cost to the user can be accurately determined in terms of CPU utilization, network throughput, and message delay. Our philosophy is that a user should not pay an overhead penalty for a service he does not use.

REFERENCES

[PROT85] Proteon, Inc., *ProNET Model p1300 IBM PC Local Network System Installation and Programming Guide*, part number 212-032, revision 3.0, August 1985.

[IEEE85] Institute of Electrical and Electronics Engineers, *IEEE Standard 802.2, Logical Link Control*, 1985.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of Sperry Marine Inc. and the Institute of Information Technology of the Virginia Center for Innovative Technology.

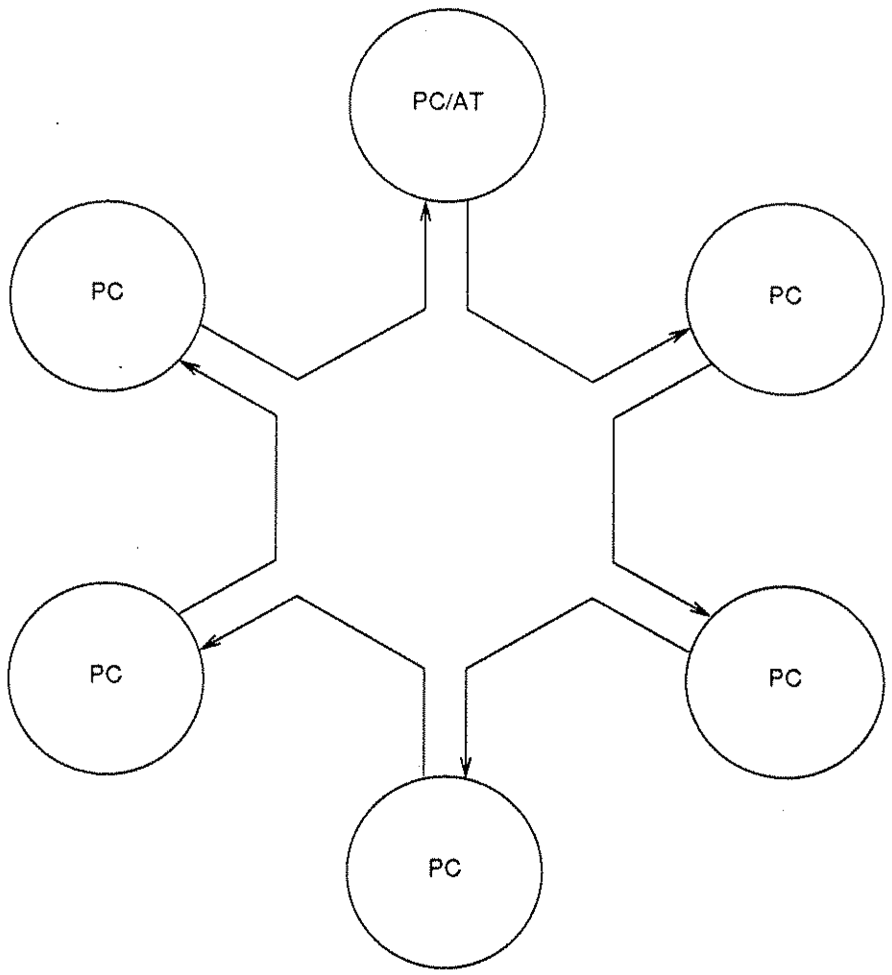


Figure 1.

Computer Networks Laboratory

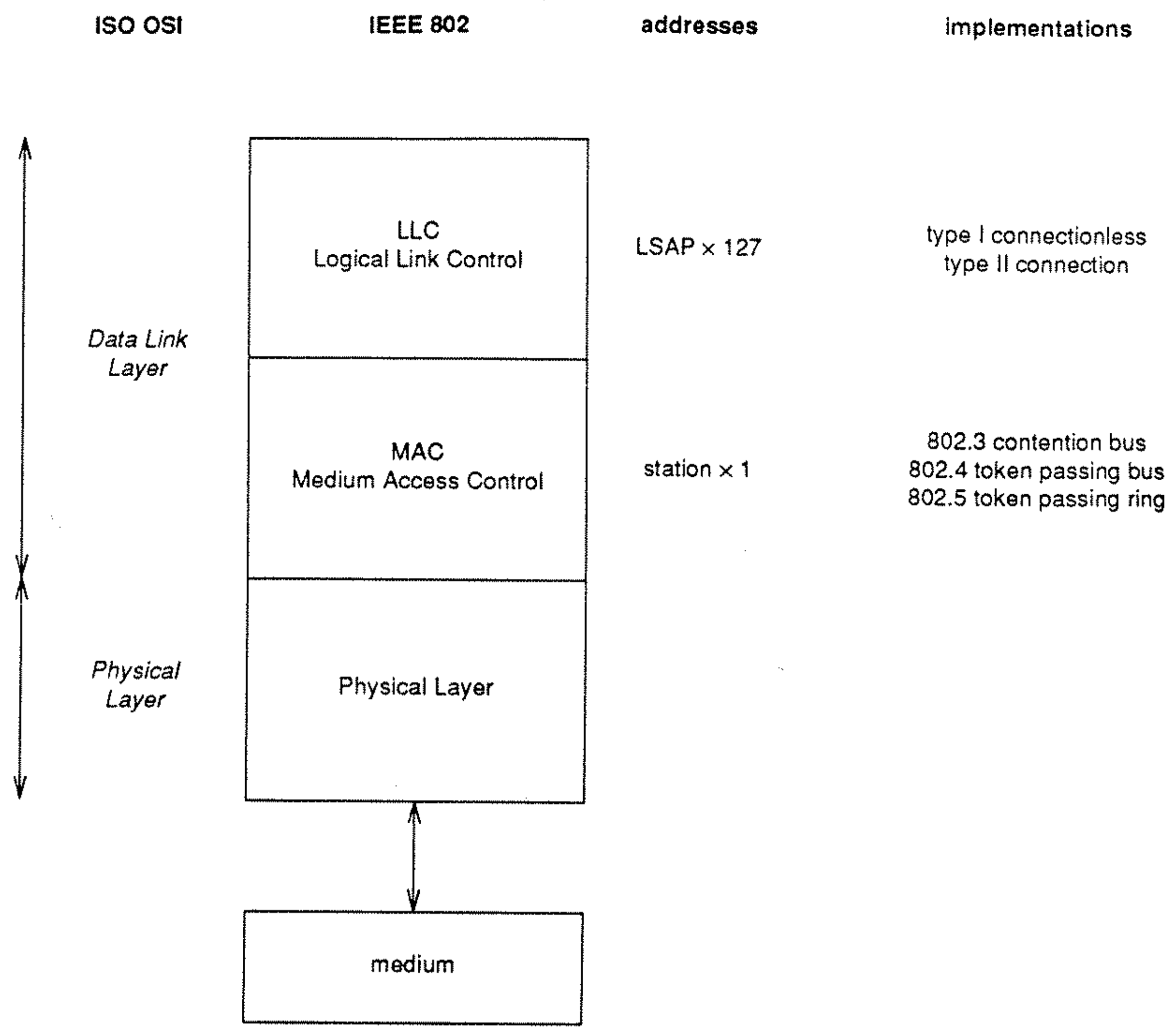
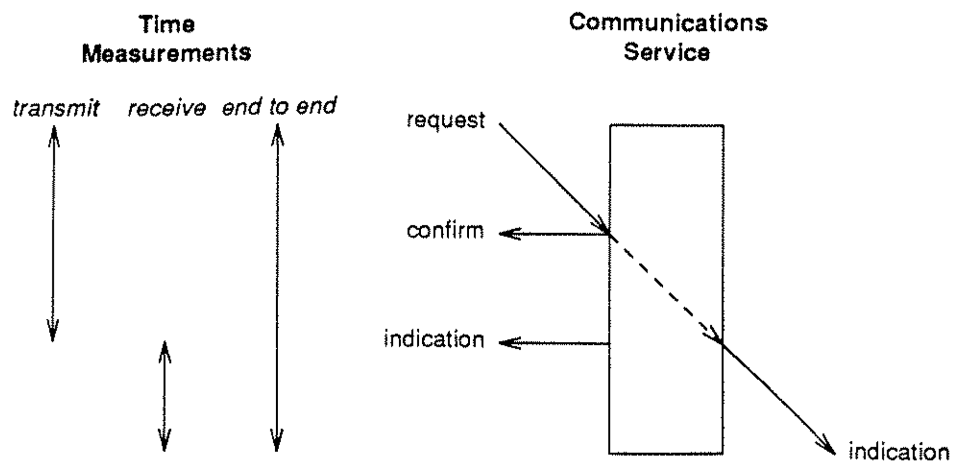
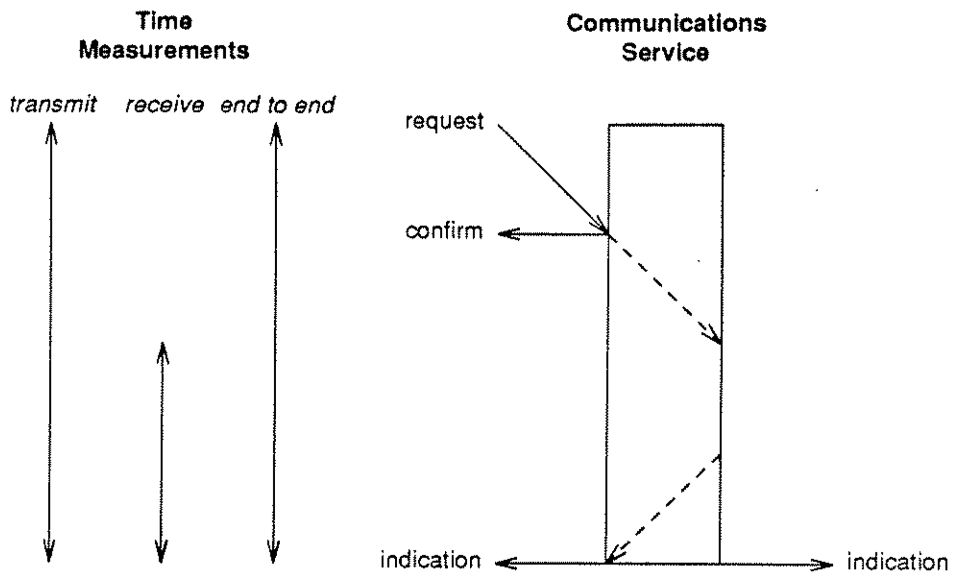


Figure 2.

OSI and IEEE Protocol Mappping



Send Data with No Acknowledge (SDN)



Send Data with Acknowledge (SDA)

Figure 3.

LLC Supported Services

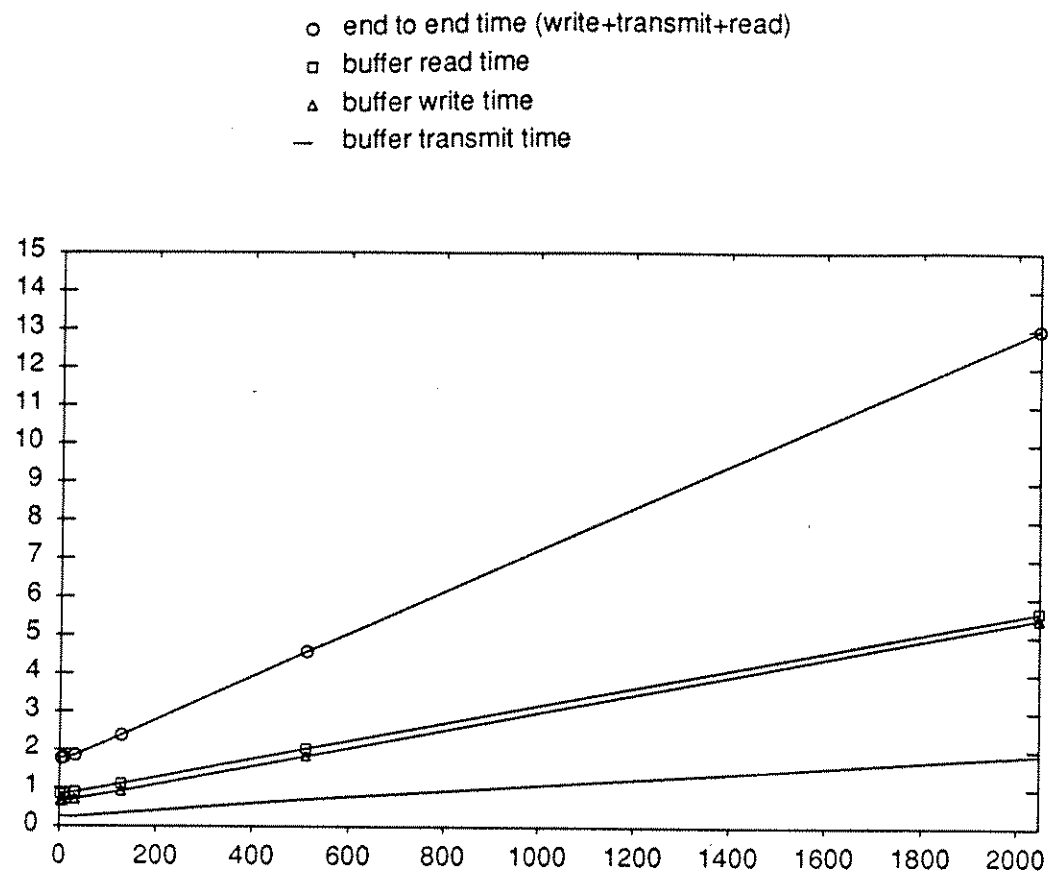


Figure 4.

MAC Operation Times (msec) *vs.* **Packet Size** (bytes)

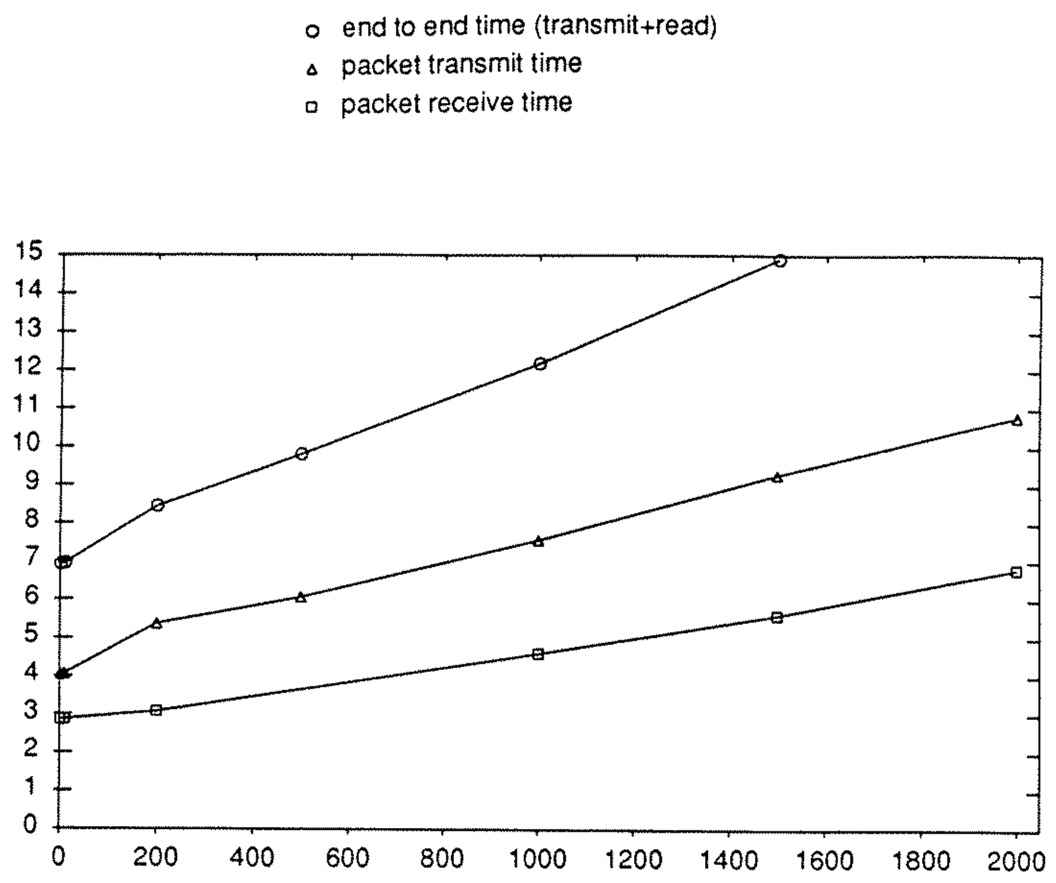


Figure 5.

LLC Operation Times (msec) *vs.* **Packet Size** (bytes)

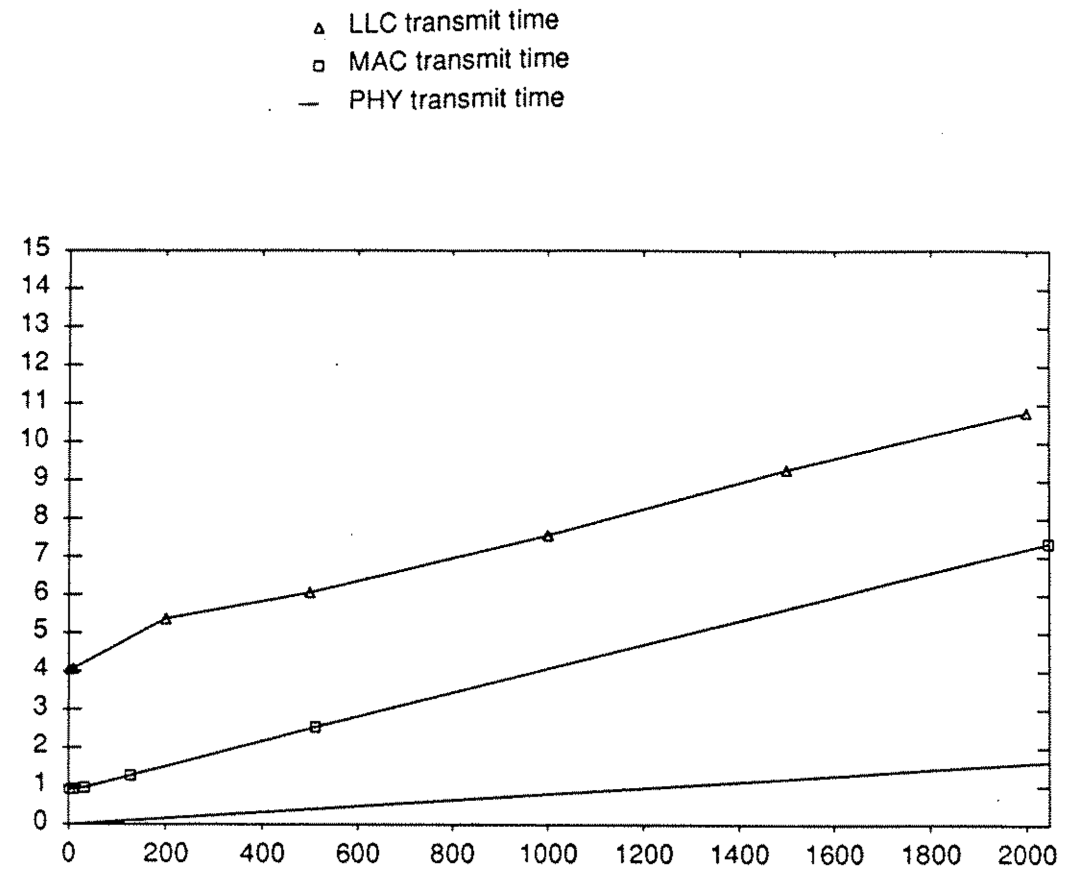


Figure 6.

Transmit Times (msec) *vs.* **Packet Size** (bytes)

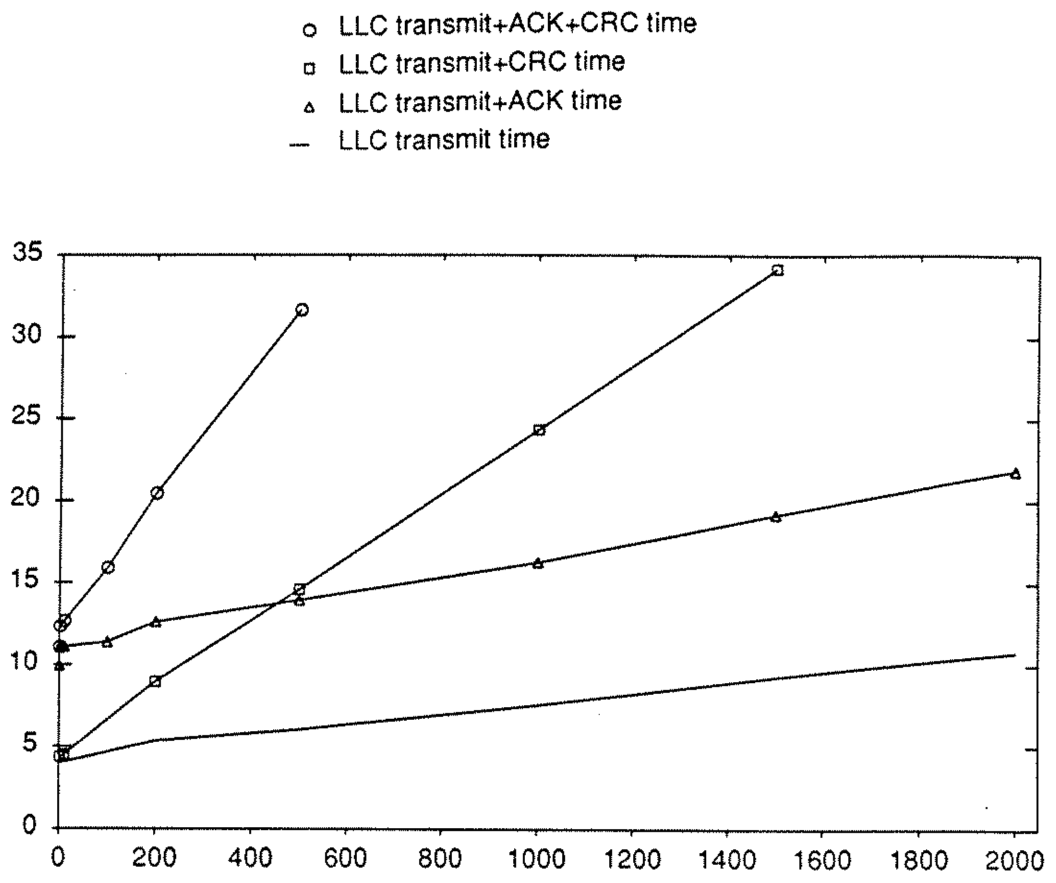


Figure 7.

LLC Transmit Option Times (msec) *vs.* **Packet Size** (bytes)

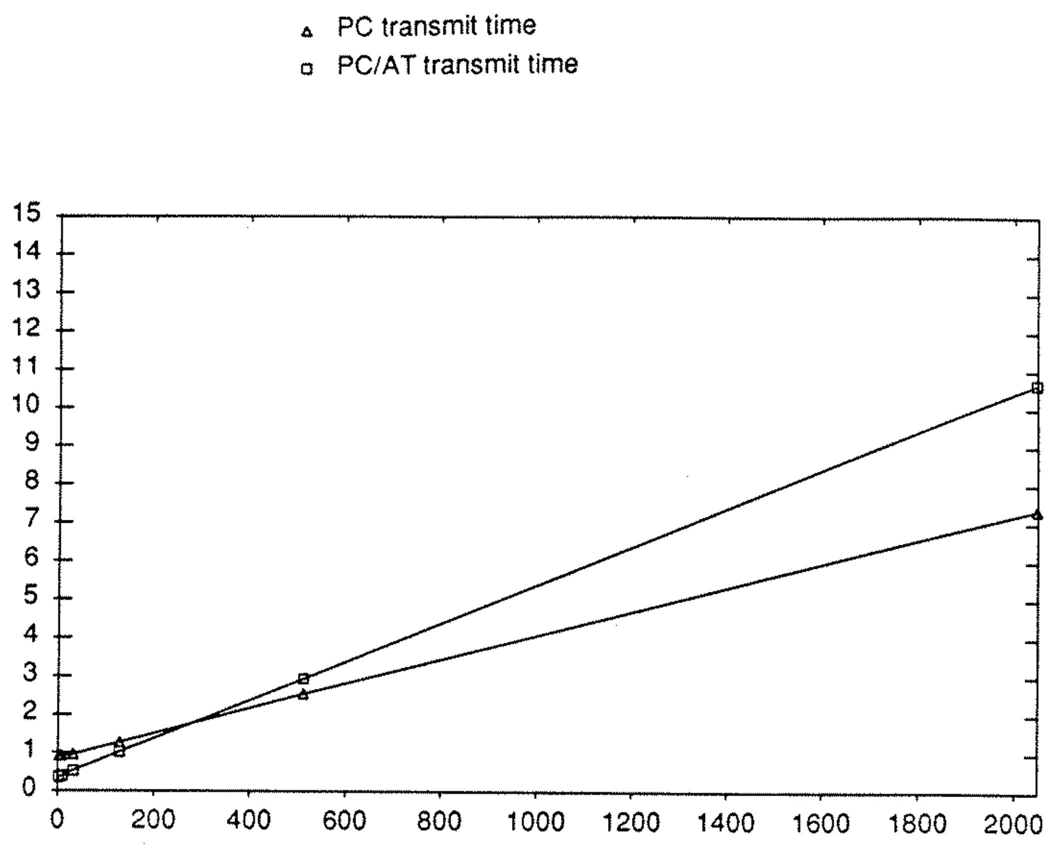


Figure 8(a).

MAC Transmit Times (msec) *vs.* **Packet Size** (bytes)

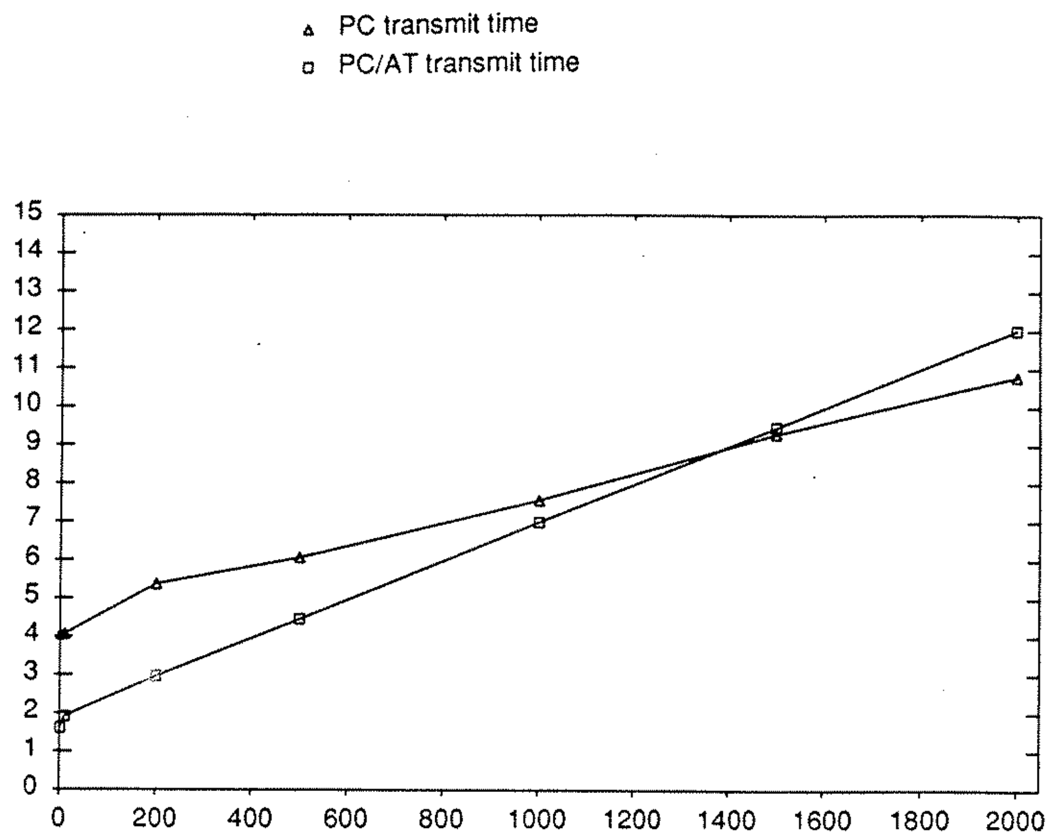


Figure 8(b).

LLC Transmit Times (msec) *vs.* **Packet Size** (bytes)

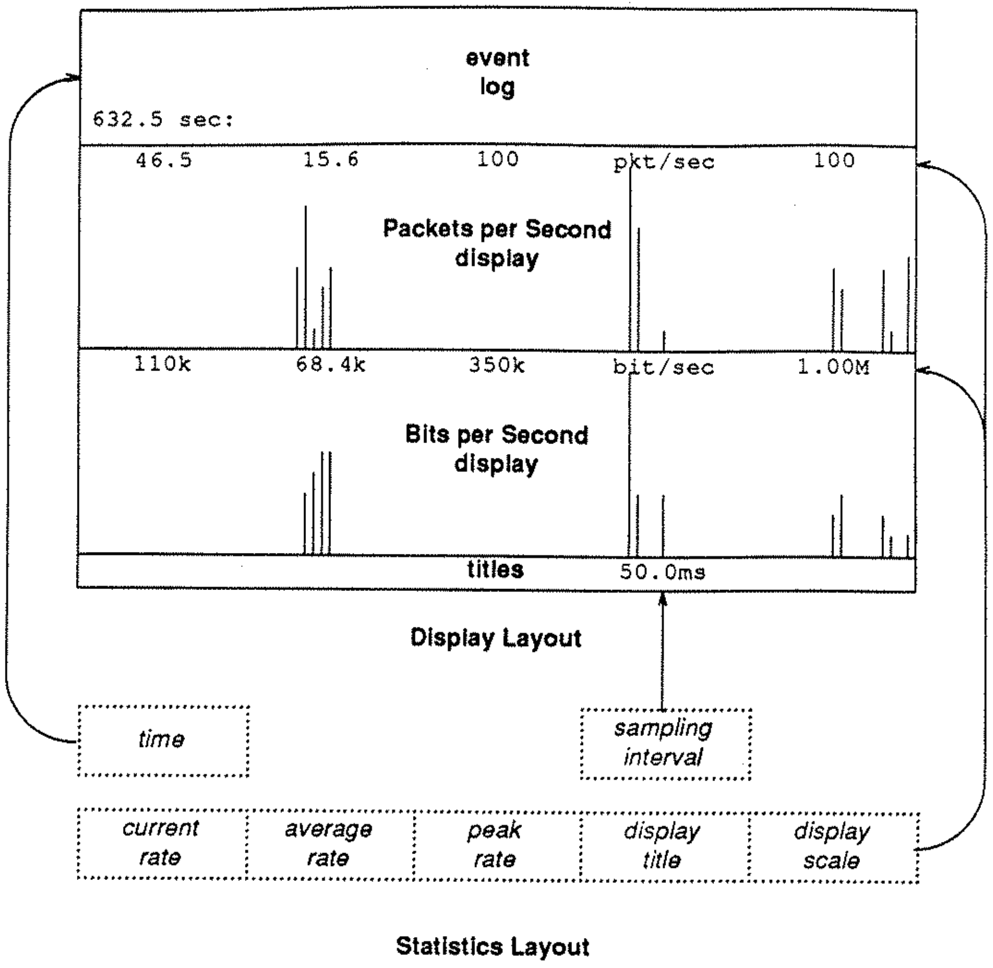


Figure 9.

Screen Display for Real-Time Network Monitor