

- 1 -

BUFFER MEMORY MANAGEMENT FOR A COMPUTER NETWORK NODE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of the networking of computer systems, in particular, buffer management between a host processor and a network adapter.

2. Prior Art

Local Area Networks (LANs) have been developed for interconnecting various data terminal equipment (DTE) such as computers, workstations and the like (hereinafter collectively referred to as workstations). One common type of LAN utilizes a shared communication channel, with access to the common channel being coordinated by the workstations themselves. This form of access control is known as Carrier Sense Multiple Access with Collision Detection (CSMA/CD). CSMA/CD access control is described in a number of publications including, for example, Tanenbaum, A.S., Computer Networks, second edition, Prentice-Hall, Inc., 1988 at pages 128-130 and also at Meijer, A., and Peeters, P., Computer Network Architectures, Computer Science Press, 1983 at pages 280-283. The Institute of Electrical and Electronics Engineers (IEEE) has provided a standard for CSMA/CD which is known as the IEEE 802.3 standard. Well known implementations of CSMA/CD include Ethernet®, DECnet and StarLan.

In general, CSMA/CD may be described as follows: a workstation wishing to transmit on the common channel first monitors the channel to determine whether another workstation is using the channel. If the channel is busy, as indicated by the presence of a carrier on the channel, the workstations wishing to transmit will defer transmission. This may be thought of as the carrier sense portion of the protocol. If the channel is not busy, the workstations will transmit a data packet over the

EXHIBIT INDEX	
1. Introduction	pages 1-10
✓ 2. Background	pages 11-27
3. Summary	pages 28-30
4. Detailed Description	pages 31-100
5. Claims	pages 101-110

channel, with the packet containing an address of the intended recipient. If a workstation detects the channel is busy or if the workstation detects that a collision has occurred on the channel, the workstation will back-off for a period of time and then again sense the channel to determine if the channel is busy.

5

In order to participate in the LAN environment, the workstation must have an adapter. The adapter provides the physical and logical interface functions between the LAN and the host processor. The adapter typically resides on the bus structure of the workstation. During the transfer of data to and from the network, the data is temporarily stored on memory associated with the adapter, e.g. via a FIFO or a buffer. In a FIFO implementation, FIFO queues for both transmitting into the network and receiving from the network must be maintained. The FIFO's are typically on-chip with the primary adapter logic. While sufficient for environments with limited network traffic, such FIFO management may degrade system performance under periods of heavy load.

15

In a buffered scheme, the memory may be off-chip to the adapter logic chip. Moreover, the size of the buffer may be greater than in the FIFO approach. For light loads, the FIFO management scheme may have performance benefits. But as noted above, during periods of heavy load, the buffer memory is a more desirable approach.

20

In known systems, transmission and receipt of data requires that an entire data packet be in memory prior to it being received by a host processor or transmitted into the network. This may introduce significant delays into both the operation of the workstations and the network. It is an object of the present

25

invention to reduce any delay introduced by the requirement of having an entire packet in buffer memory.

SUMMARY

A method and apparatus to optimize buffer memory management in a computer system communications resource is disclosed. The present invention decreases the elapsed time needed for receiving data from and transmitting data to the network. Consequently, data spends less time in the communication resource buffer memory. This allows a communications resource to handle more data without increasing the size of its buffer memory and to increase the effective data transfer rate.

Communications resources with buffer memory require that groups of data (called packets) be completely copied to the resource before starting transmits and completely received from the communications network before starting to copy data from the resource. The present invention provides a means for:

- 1) Starting the transmission of data into the network before all data has been transferred to buffer memory in the communications resource.
- 2) Beginning the transfer of data from the buffer memory of the communications resource before the complete data packet has been received and stored in buffer memory.
- 3) Error checking to insure data integrity.

With respect to data transmission, the host computer transfers an initial portion of data to the communications resource. Thereafter, a transmit command is issued which initiates the transmission of data into the network. A second transfer is initiated by the host computer to complete the transfer of data to the communication resource. The data is stored in a transmit buffer coupled to the communications resource.

The resource monitors the transfer process versus the transmit process to ensure that the correct data is available for transmit. If the transmit process tries to send data that has not yet been transferred and stored in the resource transmit
5 buffer, the resource aborts the transmit process and sets a flag to indicate to the host computer that the error has occurred. The host computer may then initiate a re-transmission of the data

With respect to receipt of the data from the network the host computer
10 calculates and writes a threshold value to the communications resource. This threshold is used by the communications resource to indicate when, during the reception of a packet, the resource will interrupt the host computer to indicate that a packet is currently being received. The host computer then may copy all or some of the data that has been received. After examining the data to determine
15 the size of the incoming data packet, the host may then wait to copy data until the entire packet is received (in the case of a small packet) or calculate a new threshold value which it writes to the resource (in the case of a large packet). The resource will generate a second interrupt to the host computer when this second threshold is reached. The host computer then begins the process of finishing the
20 copy of the remaining data from the resource. The communications resource will monitor the receipt process to determine that the data received by the host computer is valid. If it is not, a flag is set to indicate to the host computer that an error has occurred. If an error has occurred, the host computer may initiate a re-transfer of the data from the communications resource. Data incoming from the
25 network is stored in a receive buffer coupled to the communications resource.

The communications resource may typically be implemented as a Local Area Network (LAN) Adapter for a computer system, a node for a multi-processor computer system or any other system in which processing systems are in communication.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 is a block diagram of the components of a computer system with a Local Area Network (LAN) adapter coupled to a LAN as may be implemented
5 in the currently preferred embodiment of the present invention.

Figure 2 is a block diagram illustrating the major functional components of a LAN adapter as may be utilized in the currently preferred embodiment of the present invention.
10

Figures 3a and 3b are flow charts which illustrate the steps performed by the host processor and the adapter for early receipt of data as been performed in the currently preferred embodiment of the present invention.

Figure 4 is a flowchart illustrating the steps performed by the host and the adapter for early transmittal of a data packet in the currently preferred
15 embodiment of the present invention.

Figure 5A is a timing diagram illustrating the receipt of data in a typical
20 prior art system.

Figure 5B is a timing diagram illustrating the early receipt of data as may be performed in the currently preferred embodiment of the present invention.

Figure 6A is a timing diagram illustrating the transmittal of a data packet in a typical prior art system.
25

Figure 6B is a timing diagram illustrating the timing sequence for the early transmittal of a data packet in the currently preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for optimization of buffer memory management in a communications resource is described. In the following description, numerous specific details are set forth such as the timing of data signals in order to provide a
5 thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without such specific details. In other instances, specific implementation details such as the physical interface to the Local Area Network (LAN) media or a CPU bus have not been shown in detail in order not to unnecessarily obscure the present invention.

10 While the communications resource of the currently preferred embodiment of the present invention is implemented as a LAN interface adapter supported by a general purpose computer, it would be apparent to one skilled in the art to embody the present invention in a fixed function system, e.g. an X-Terminal. Further, the present invention may be embodied as an adapter to couple to a computer system
15 into any type of environment where processing systems are in communication, e.g. a parallel processing system. Such embodiments would not depart from the spirit and scope of the present invention.

The computer system 100 of the currently preferred embodiment is described with reference to Figure 1. A computer system as may be utilized by the preferred
20 embodiment generally comprises a bus structure or other communication means 101 for communicating information between the various components of the computer system, a processor means 102 coupled with said bus 101 for processing information, a random access memory (RAM) or other storage device 103 (commonly referred to as a main memory) coupled with said bus 101 for storing information and
25 instructions for said processor 102, a read only memory (ROM) or other static

storage device 104 coupled with said bus 101 for storing static information and instructions for said processor 102, a display monitor 105 is coupled with said bus 101 for communication with the processor 102, a data storage device 106, such as a magnetic disk or disk drive, coupled with said bus 101 for storing information and instructions, an alphanumeric input device 107 including alphanumeric and other keys coupled to said bus 101 for communicating information and command selections to said processor 102, a cursor control device 108, such as a mouse, track-ball, cursor control keys, etc., coupled to said bus 101 for communicating information and command selections to said processor 102 and for controlling cursor movement. Additionally, the system will typically include, one or more ports for receiving input signal data. Such ports are illustrated here as serial port 108 and parallel port 109.

The computer system 101 further includes a Local Area Network (LAN) adapter 111 for attaching to a LAN 112. The LAN Adapter 111 is coupled to CPU bus 101 and is used to transmit/receive information to/from the LAN 112. The LAN adapter is further described with reference to Figure 2. In the currently preferred embodiment of the present invention, the LAN 112 is an Ethernet LAN.

It should be noted that the processor 102 will perform functions that are hereinafter referred to as Host functions. Such functions will become apparent in the description of that follows. Further, RAM 103 and data storage device 106 are used to handle and store the data packets to be transmitted to or received from the network.

Referring to Figure 2, the LAN adapter is controlled by control logic module 201. Typically, the control logic 201 may be part of an ASIC chip or a specifically designed integrated circuit chip. In any event, the control logic 201

controls the receipt, and transmittal of data to and from the LAN. A CPU bus interface 202 provides the physical interface to the processor via the processor bus. The LAN interface 203 provides the physical connections for interfacing to the LAN. A Receive Buffer Area (RBA) 204 is used to store data packets received from the network. The Transmit Buffer Area (TBA) 205 is used to store data that will be transmitted from the host processor to the network.

In known implementations, before a data packet may be transmitted or received, the entire packet must be in either the RBA 204, for receiving data, or the TBA 205, for transmitting data into the network. Clearly this does not make optimal use of either the computer system bus or the LAN. For example, if a large data packet is being received, the entire data packet must be received and stored in the receive buffer area prior to it being transmitted to the appropriate location on the computer system via the bus. If the receipt of the data from the network and the transfer of the data to the location in the computer system could be overlapped, an aggregate amount of time from when the data is first received by the computer system to the time at which it is stored on the computer system may be significantly reduced. Similarly, for the transmission of data into the network, the amount of time can be significantly reduced from the initiation of the transfer of data to when the data is actually on the network could be reduced if the transfer time and the transmit time were overlapped. In both instances, the currently preferred embodiment of the present invention provides for such overlap.

The control logic module 201 in the LAN adapter will support early interrupt generation during packet reception. This means that the interrupt may occur any time after data is placed in the receive buffer. The present invention provides early interrupts to the Host (e.g. the processor 102 of Figure 1) based upon certain events.

These events are observed as buffer accesses into the Receive Buffer Area (RBA). In the currently preferred embodiment, the event monitored is the writing of data into the RBA. A counter counts the number of bytes written into the RBA. When a predetermined number of bytes have been written into the RBA (i.e. the threshold),
5 the early receive interrupt is generated. The predetermined number of bytes would be generated by the host dependent on such facts as the Network Operating System and LAN topology being utilized.

In an alternative embodiment, the event monitored is the writing to a specific address in the RBA. In such an alternative embodiment, again the host would
10 generate the specific addresses in the RBA to be monitored. Thus, in the alternative embodiment, early interrupts are generated when accesses to particular RBA address locations are detected. Preferably, in the alternative embodiment two independent, programmable interrupt trigger/threshold addresses are provided. Each trigger/threshold address may be either a receive data location, or a
15 status/command location. Use of both provides flexibility across different frame types, packet sizes, etc. .

The present invention may also monitor write accesses to the RBA. The result is a word count indicating the amount of data currently available for transfer by the host, even though packet reception may not be complete. To support this word
20 count function, the driver must provide the start and end address of the memory area to be monitored (i.e., the current portion of the RBA). The word count is incremented by one when the communications resource writes to the area delimited by these boundaries. If the packet size is an odd number of bytes, the last data write will actually be a single byte, and this final count will exceed the true count by one.
25 Thus this count is intended for use only while moving early data. A separate count field is referred to upon completion of the entire data packet.

In case of packet transmission, a performance increase may be realized by setting the communications resource transmit command active before the entire packet has been downloaded to the buffer from system memory (e.g. RAM 103 of Figure 1). Unfortunately, this model has the potential for failure if the system is
5 unable to stay ahead of the communications resource in the use of the transmit data buffer. If this were to happen, the communications resource would read the incorrect data, append a valid CRC, and give no indication of an error.

Although the host interface is faster than the communications resource requirements, the driver could fall behind if it is interrupted or if bus master devices
10 consume excessive bus bandwidth. While it is possible to disable interrupts during this period, it is not possible for the driver to prevent bus masters from acquiring the bus. In addition, disabling interrupts for a long period of time may have an adverse effect on other interrupt driven devices, such as serial port modems. The most compatible approach is to leave interrupts enabled and provide a mechanism for
15 detecting an underrun condition.

The communications resource can detect the underrun condition when the Host write pointer equals the communications address resource. Detection may be controlled by an enable flag which must be set by the host before attempting the early transmit, and cleared by the host when complete. When an underrun is
20 detected, the communications resource will set an error flag to set in motion the creation of an erred packet (which is described in greater detail below).

The transmit routine may read the error flag after moving each block, or after moving the entire packet. If an underrun occurred, the driver need only reissue the transmit command, because there will have been no corruption of the transmit
25 buffer.

A more detailed description of the procedures for early receipt and early transmittal now follow.

Early Receive

Figures 3a and 3b are flowcharts which illustrate the step performed by
5 the host processor and the adapter for early receipt of a data packet. First, the host sets a first receive threshold and transmits this value to the adapter, step 301. The first data receive threshold is typically a number of bytes comprising the packet header information. The value of the first data threshold is dependent on such factors as the Network Operating System being used as well as the
10 physical LAN implementation. For example, in the 802.3 format, the header information will be received in the first fourteen (14) bytes. So by setting the first data threshold to 14 bytes it can assured that the packet header is within the Receive Buffer Area (RBA) of the LAN Adapter. Typically, this data is sent to the adapter once and retrieved for later use. In any event, the adapter will then
15 receive data from the network and store it in the RBA, step 302. During the receive and store process a check is made to determine whether or not the first threshold is hit, step 303. When the first threshold is hit, a first threshold interrupt is generated and transmitted to the host processor, step 304. When the host processor receives this interrupt it processes it by transferring this initial
20 amount of data from the RBA, step 305. As noted above this information will comprise the packet header. The size of the packet is then determined, step 306. The packet will typically contain the packet size as well as certain addressing information. At this point, the packet size information is examined. A check is then made to determine whether the packet is small, step 307. If the packet is
25 small, e.g. sixty-four (64) bytes, it may not be worthwhile to try and receive data early. If the packet is small, the host processor may allow the entire data packet

to be received, before transferring any of the data. Thus, the adapter will receive and store data into the RBA from the network, step 308 and check for the end of the packet, step 309. So long that it is not at the end of the packet the adapter will continue the receiving and storing of data. When all the data in the packet is
5 received an end of packet interrupt is generated, step 310. When the end of pack interrupt is received by the host, it will copy the remainder of the packet data from the RBA, step 311. Once the entire data packet is received by the host it will commence protocol error checking, step 312. This step is comprised of the normal processing for the protocol being used.

10

Assuming now that the packet is not small, a second threshold value is generated and transmitted to the adapter step, 313. As noted above, in the preferred embodiment the second threshold value is a predetermined byte count and in an alternative embodiment it is a specific address in the RBA. Referring
15 now to Figure 3b, the adapter will again continue to receive data from the network and store the received data in the RBA, step 314, until the second threshold is hit, step 315. When the second threshold is hit a second threshold interrupt is generated and transmitted to the host, step 316. When the second threshold interrupt is received by the host it will commence copying data from
20 the RBA, step 317. Concurrent with the transfer of data from the adapter, the adapter will receive and store data from the network, step 318 and transfer that data to the host, step 319. During the receipt and storing of data from the network and transferring from the data to the host, overrun conditions are checked for, step 320. An overrun condition will occur when the data which is
25 being transferred from the adapter receive buffer memory to the host exceeds the data being received from the network. Irrespective of an overrun condition being detected, the processing and receipt of data will commence until the end of

packet is reached, step 322. This is to insure that the receipt of the entire data packet occurs into the RBA. When the end of packet is received an end of packet interrupt is generated and transferred into the host, step 323. When the end of packet interrupt is received by the host, a check is made to determine whether or not the error flag is set, step 324. If the error flag has been set, the host will
5 reinitiate transfer of the data from the RBA, step 325. In this case, all the data will have to be transferred. If the error flag is not set, the remainder of the data in the RBA is transferred, step 326. In either a circumstance, once the data has been received from the RBA, the protocol error checking is performed, step 327.

10 Referring to Figure 5A, a timing diagram illustrates the receipt of data in a typical prior art system. As indicated in Figure 5A, network data is received between timing edges 516 and 518. Subsequent to the receipt of network data, an end of packet interrupt 522 occurs and the host computer begins to read in the data packet at edge 520. Note that the reception of data from the network and
15 the reading of the data by the host computer is performed serially in the prior art.

Referring now to Figure 5B, a timing diagram of the early data receive process of the present invention is illustrated. In the present invention, while network data 554 is being received from the network, a second threshold interrupt 566 occurs. At this point, the host computer begins to read data at edge
20 568. The host data read occurs concurrently with data coming in from the network. Thus, once the host completes the data read at edge 570, the present invention has realized a decrease in the amount of time required to receive data in comparison to prior art systems as indicated by line 580 in Figure 5B.

Early Transmittal

25 Figure 4 is a flowchart describing the steps performed by the host and the adapter for the early transmission of data. Referring to Figure 4, first a

predetermined amount of data is transferred from the host to the Transmit Buffer Area (TBA) of the LAN adapter, step 401. Typically, this predetermined amount of data will include the packet header plus some other predetermined number of bytes. Next, the adapter will receive and store the data from the host in the TBA, 5 step 402. The host processor will then issue a transmit command to transfer the data from the TBA in the LAN adapter to a desired location within the network, step 403. Thereafter, a transfer command will be issued by the host to transfer the remainder of the data to the TBA of the adapter, step 404. Concurrently therewith data is transmitted onto the network by the adapter, step 405, and data 10 is received from the host and stored in the TBA, step 406. During this transfer and transmittal of data, an underrun condition is checked for, step 407. An underrun condition will exist if the data being transmitted on to the network is not "behind" the data being transferred from the host to the adapter TBA. This will occur when the address of the adapter TBA that is accessed to transmit data 15 to the network exceeds the address to which data is being transferred to the TBA. If an underrun condition exists, an error flag is set and the adapter is set-up to cause subsequent packet alteration to insure the recipient knows the transmitted data is invalid, step 408. The packet being transmitted will further be altered to insure that the data packet transmission will be recognized by a recipient as 20 containing errors. The manner in which the packet is altered to indicate invalid data is dependent on the transmission protocol being utilized. For example in the 802.5 CSMA/CD protocol the encoded CRC field will cause the recipient to invalidate the packet. In any event, whether or not an underrun condition exists the data is transferred to the TBA until the data transmission is completed, step 25 409. When the data transmission is complete, the host will check to determine whether or not the error flag had been set, step 410. If the error flag has been set the host will issue a transmit command to retransmit the data in the TBA. At this

point the adapter data would then transmit the data onto the network. If the error flag was not set than the host processor will recognize that the transmission was without error as a result of early transmittal.

Figure 6A is a timing diagram illustrating the transmittal of a data packet in a typical prior art system. In a typical prior art implementation, a host computer writes data to a buffer between timing edges 614 and 616. Subsequently, data is transmitted from the buffer on the network between edges 618 and 620. Note that the host write and network transmission operations occur serially in the prior art.

Referring now to Figure 6B, a timing diagram illustrates the timing sequence for the early transmittal of data in the preferred embodiment of the present invention. As shown in Figure 6B, the host processor issues a transmit command during timing edges 654 and 656. Subsequently, the host processor writes a data packet between edges 656 and 660 while data is transmitted on the network between timing edges 658 and 662. The host write of the data packet occurs concurrently with the data being transmitted on the network. Thus, the present invention realizes a decrease in the transmit time required for transmitting data over the network in relation to the prior art. This decrease in the transmit time is indicated in Figure 6B by Line 664.

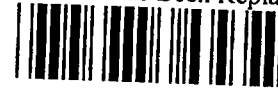
20

Thus, a Buffer Memory Management Module for a Computer Network Node which provides for early receipt and transmittal of packets, is disclosed.

25



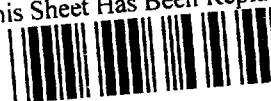
This Sheet Has Been Replaced



-19-

CLAIMS

- 5 1. An interface for coupling a computer system to a network for transmitting and receiving data packets to and from said network, said interface comprising an adapter portion and a host portion, said adapter portion comprising:
- a data packet transmission buffer;
 - a data packet receiving buffer;
- 10 first error detection means for detecting that a first data packet being transmitted from said data packet transmission buffer contains invalid data and setting a first error flag;
- second error detection means for detecting that a second data packet being transferred from said data packet receiving buffer contains invalid data and
- 15 setting a second error flag;
- threshold interrupt generation means for generating a receive interrupt signal to said host portion when a predetermined number of bytes are received in said data packet receiving buffer;
- said host portion comprising:
- 20 packet transmission command generation means for issuing a plurality of commands in a predetermined sequence to cause transmission of said first data packet onto said network prior to the entirety of said first data packet being transferred to said data packet transmission buffer of said adapter portion;
- first error handling means for detecting said first error flag is set and
- 25 issuing a transmit command to said adapter portion to transmit said first data packet in said data packet transmission buffer;



packet receiving means for receiving portions of a data packet from said data packet receiving buffer responsive to said receive packet interrupt signal, prior to an entire data packet being in said packet transmission buffer; and

second error handling means for detecting said second error flag is set and
5 issuing a transfer command to transfer said data packet in said second data packet receiving buffer.

2. The interface as recited in Claim 1 wherein said plurality of commands in a predetermined sequence issued by said packet transmission command
10 generation means is comprised of a first command to transfer a first portion of said data packet to said data packet transmission buffer, a second instruction to cause said data packet to be transmitted from said data packet transmission buffer, and a third instruction for transferring a second portion of said data packet to said data packet transmission buffer.

15

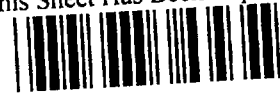
3. The interface as recited in Claim 2 wherein said packet receiving means is further comprised of:

data packet header receiving means for receiving a data packet header of said second data packet and determining the size of said second data packet; and

20 threshold generation means for providing said threshold interrupt generation means with a threshold value if said second data packet size exceeds a predetermined size.

4. The interface as recited in Claim 3 wherein said first error detection means
25 is further comprised of means for determining that data being transmitted onto said network is not part of said first data packet.





5. The interface as recited in Claim 4 wherein said means for determining that data being transmitted onto said network is not part of said first data packet is further comprised of a first comparator for comparing a first address where data is being transferred to said data packet transmission buffer and a second
5 address where data is being transmitted from said data packet transmission buffer.
6. The interface as recited in Claim 4 wherein said second error detection means is further comprised of means for determining that data being transferred
10 from said data packet receiving buffer is not part of said second data packet.
7. The interface as recited in Claim 6 wherein said means for determining that data being transferred from said data packet receiving buffer is not part of said second data packet is further comprised of a second comparator for
15 comparing a third address where data is being received by said data packet receiving buffer and a fourth address where data is being transferred from said data packet receiving buffer.
8. A method for receiving a data packet from a network and transferring
20 said data packet to a storage medium on a computer system, said method comprising the steps of:
- a) detecting that a first portion of said data packet has been received and stored in said data packet receiving buffer;
 - b) transferring said first portion of said data packet to said storage
25 medium;
 - c) examining said first portion of said data packet to determine a size of the data packet;



- d) if said data packet is less than or equal to a predetermined size, waiting until the entire data packet is received and stored in said data packet receiving buffer before transferring said data packet into said storage medium;
- e) if said data packet is greater than said predetermined size
- 5 generating a threshold value representing the amount of data of said data packet that is to be received and stored in said data packet receiving buffer prior to transferring to said storage medium;
- f) detecting that said threshold value has been reached;
- g) initiating the transfer of data from said data packet receiving buffer
- 10 to said storage medium;
- h) determining if the data being transferred to said storage medium is valid or invalid;
- i) if said data being transferred is invalid, setting an error flag;
- j) checking if said error flag has been set; and
- 15 k) if said error flag has been set, re-transferring said data packet from said data packet receiving buffer to said storage medium.

9. The method as recited in Claim 8 wherein said step of determining if the data being transferred to said storage medium is valid or invalid is further
- 20 comprised of the steps of:
- comparing a first address of said data packet receiving buffer where data is being received with a second address of said data packet receiving buffer from which data is being transferred to said storage medium;
 - if said second address does not exceed an address range where said data
 - 25 packet has been received, said address range defined by said first address, identifying said data being transferred as valid; and



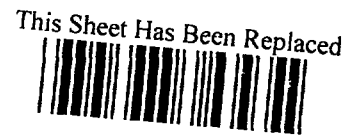
This Sheet Has Been Replaced



-23-

if said second address exceeds said address range where said data packet has been received, identifying said data being transferred as invalid.

10. The method as recited in Claim 9 wherein said first portion of said data
5 packet comprises a data packet header.
11. The method as recited in Claim 10 wherein said predetermined size is
sixty-four bytes.
- 10 12. The method as recited in Claim 11 wherein said first portion is fourteen
bytes in length.
13. A method for transmitting a data packet onto a network from a computer
system, said method comprising the steps of:
 - 15 a) transferring a first portion of said data packet to a data packet
transmission buffer;
 - b) commencing the transmission of said data packet to said network
from said data packet transmission buffer;
 - c) transferring the remainder of said data packet to said data packet
20 transmission buffer;
 - d) determining if said data being transmitted from said data packet
buffer is valid or invalid; and
 - e) if said data being transmitted is invalid, re-transmitting said data
packet from said data packet transmission buffer.



-24-

14. The method as recited in Claim 13 is further comprised of the step of if said data being transmitted is invalid, causing said data packet to contain information identifying said data packet as invalid.

5 15. The method as recited in Claim 14 wherein said step of determining if said data being transmitted from said data packet buffer is valid or invalid is further comprised of the steps of:

comparing a first address of said data packet transmission buffer to where data is being transferred with a second address of said data packet transmission
10 buffer from which data is being transmitted to said network;

if said second address does not exceed an address range where said data packet has been transferred, said address range defined by said first address, identifying said data being transmitted as valid; and

if said second address exceeds said address range where said data packet
15 has been transferred, identifying said data being transmitted as invalid.

16. The method as recited in Claim 15 wherein said first portion is comprised of fourteen bytes.

20 17. A computer system comprising:

a processor;

a bus;

one or more memory devices;

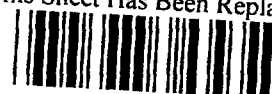
one or more data input devices;

25 one or more data output devices;

one or more storage devices;



This Sheet Has Been Replaced



-25-

a network interface for interfacing into a network, said network interface comprising an adapter portion and a host portion, said adapter portion comprising:

a data packet transmission buffer;

5 a data packet receiving buffer;

first error detection means for detecting that a first data packet being transmitted from said data packet transmission buffer contains invalid data and setting a first error flag;

10 second error detection means for detecting that a second data packet being transferred from said data packet receiving buffer contains invalid data and setting a second error flag;

threshold interrupt generation means for generating a receive interrupt signal to said host portion when a predetermined number of bytes are received in said data packet receiving buffer;

15 said host portion comprising:

packet transmission command generation means for issuing a plurality of commands in a predetermined sequence to cause transmission of said first data packet onto said network prior to the entirety of said first data packet being transferred to said data packet transmission buffer of said adapter portion;

20 first error handling means for detecting said first error flag is set and issuing a transmit command to said adapter portion to transmit said first data packet in said data packet transmission buffer;

packet receiving means for receiving portions of a data packet from said data packet receiving buffer responsive to said receive packet interrupt signal, 25 prior to an entire data packet being in said packet transmission buffer; and



This Sheet Has Been Replaced



-26-

second error handling means for detecting said second error flag is set and issuing a transfer command to transfer said data packet in said second data packet receiving buffer.

5 18. The computer system as recited in Claim 17 wherein said plurality of commands in a predetermined sequence issued by said packet transmission command generation means of said network interface is comprised of a first command to transfer a first portion of said data packet to said data packet transmission buffer, a second instruction to cause said data packet to be
10 transmitted from said data packet transmission buffer, and a third instruction for transferring a second portion of said data packet to said data packet transmission buffer.

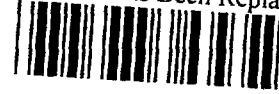
19. The computer system as recited in Claim 18 wherein said packet receiving
15 means of said network interface is further comprised of:

data packet header receiving means for receiving a data packet header of said second data packet and determining the size of said second data packet; and
threshold generation means for providing said threshold interrupt generation means with a threshold value if said second data packet size exceeds
20 a predetermined size.

20. The computer system as recited in Claim 19 wherein said first error detection means of said network interface is further comprised of a first comparator for comparing a first address where data is being transferred to said
25 data packet transmission buffer and a second address where data is being transmitted from said data packet transmission buffer.



This Sheet Has Been Replaced



-27-

21. The computer system as recited in Claim 20 wherein said second error detection means of said network interface is further comprised of a second comparator for comparing a third address where data is being received by said data packet receiving buffer and a fourth address where data is being transferred
- 5 from said data packet receiving buffer.
22. An interface for coupling a computer system to a network for transmitting and receiving data packets to and from said network, substantially as hereinbefore described with reference to the accompanying drawings.
23. A method for receiving a data packet from a network and transferring said data packet to a storage medium on a computer system, substantially as hereinbefore described.