U.S. Patent No. 7,673,072 (072 Patent)

IPR2017-1406 (Intel)
IPR2018-0375 (Dell)
IPR2017-1707 (Cavium)
IPR2018-0329 (Wistron)

*All citations herein are to the IPR2017-01406 case unless otherwise noted.

072 Patent: Instituted Grounds

- **Erickson** in view of **Tanenbaum96**
  - 072 Patent: Claims 1, 2-8, 9, 10-14 and 15, 16-21

Ex. 1005 – U.S. Patent No. 5,768,618 (Erickson)
Ex. 1006 – Tanenbaum, Andrew S., Computer Networks (Tanenbaum96)

072 Patent: Disputes

1. A POSA would have been motivated to combine Tanenbaum96 with Erickson

2. Erickson in view of Tanenbaum96 discloses the limitations of claims 1-21 of the 072 Patent

3. Motion to Amend 072 Patent should be denied

072 Patent: Disputes

1. **A POSA would have been motivated to combine Tanenbaum96 with Erickson**

- See 036 Patent, Dispute 1, slides 6-53

072 Patent: Disputes

2. The combination of Erickson and Tanenbaum96 discloses the limitations of claims 1-21 of the 072 Patent

a) **<u>The prior art discloses "dividing, by the interface device, the data into segments" (all claims)</u>**

b) The prior art discloses "transferring status information for the context to the interface device during the same operation as transferring protocol header information to the interface device" (claim 2)

c) The prior art discloses "receiving, by the interface device, receive packets that correspond to the [context/protocol information], and updating the [context/status information] by the interface device to account for the receive packets" (claims 7, 14, 21)

“Dividing, by the interface device, the data into segments”

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,673,072 B2**
(45) **Date of Patent: Mar. 2, 2010**

(54) **FAST-PATH APPARATUS FOR TRANSMITTING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 435 days.

(21) Appl. No.: **11/821,820**

(22) Filed: **Jun. 25, 2007**

(65) **Prior Publication Data**

US 2008/0126553 A1 May 29, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/260,112, filed on Sep. 27, 2002, now Pat. No. 7,237,036, which is a continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a continuation-in-part of application No. 10/023,240, filed on Dec. 17, 2001, now Pat. No. 6,965,941, which is a continuation-in-part of application No. 09/464,283, filed on Dec. 15, 1999, now Pat. No. 6,427,173, which is a continuation-in-part of application No. 09/439,603, filed on Nov. 12, 1999, now Pat. No. 6,247,060, which is a continuation-in-part of application No. 09/067,544, filed on Apr. 27, 1998, now Pat. No. 6,226,680, said application No. 10/260,112 is a continuation-in-part of application No. 09/384,792, filed on Aug. 27, 1999, now Pat. No. 6,434,620, which is a continuation-in-part of application No. 09/141,713, filed on Aug. 28, 1998, now Pat. No. 6,389,479, said application No. 10/260,112 is a conti part of application No. 09/970,124, filed 2001, now Pat. No. 7,124,205, and a conti part of application No. 09/855,979, filed 2001, now Pat. No. 7,133,940, and a conti part of application No. 09/802,550, filed 2001, now Pat. No. 6,658,480, and a conti part of application No. 09/802,426, filed 2001, now Pat. No. 7,042,898, and a conti part of application No. 09/802,551, filed 2001, now Pat. No. 7,076,568, and a conti part of application No. 09/801,488, filed 2001, now Pat. No. 6,687,758, and a conti part of application No. 09/789,366, filed 2001, now Pat. No. 6,757,746, and a conti part of application No. 09/675,700, filed 2000, and a continuation-in-part of appl 09/675,484, filed on Sep. 29, 2000, no 6,807,581, and a continuation-in-part of No. 09/514,425, filed on Feb. 28, 2000, n 6,427,171, and a continuation-in-part of No. 09/416,925, filed on Oct. 13, 1999, n 6,470,415.

(60) Provisional application No. 60/061,809, fi 14, 1997, provisional application No. filed on Aug. 27, 1998.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

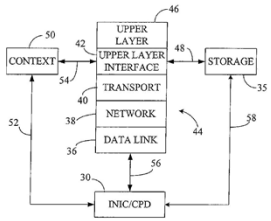
(52) **U.S. Cl.** **709/245**; 709/23 709/234; 709/232; 709/239; 370/23 370/237; 370/23

(58) **Field of Classification Search** 709/230–234, 239, 250; 370/23 370/23
See application file for complete search hi

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al. .....
4,485,455	A	11/1984	Boone et al. .....
4,485,460	A	11/1984	Stambaugh .....
4,589,063	A	5/1986	Shah et al. .....



1. A method comprising:

establishing, at a host computer, a transport layer connection, including creating a context that includes protocol header information for the connection;

transferring the protocol header information to an interface device;

transferring data from the network host to the interface device, after transferring the protocol header information to the interface device;

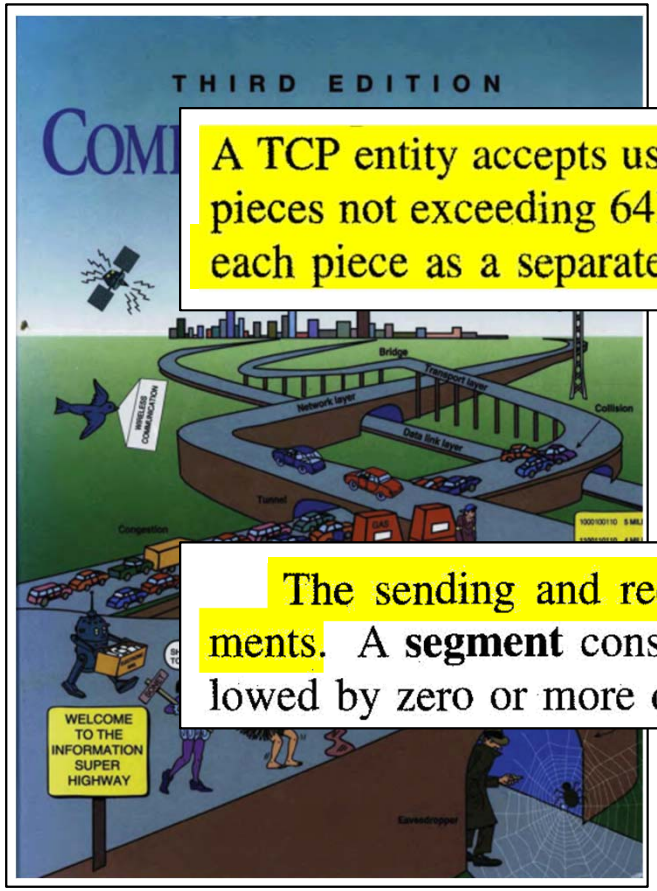
dividing, by the interface device, the data into segments;

creating headers for the segments, by the interface device, from a template header containing the protocol header information; and

prepending the headers to the segments to form transmit packets.

Ex. 1001 (072 Patent), Claim 1.

Tanenbaum96: TCP entity divides data into segments (TCP packets)



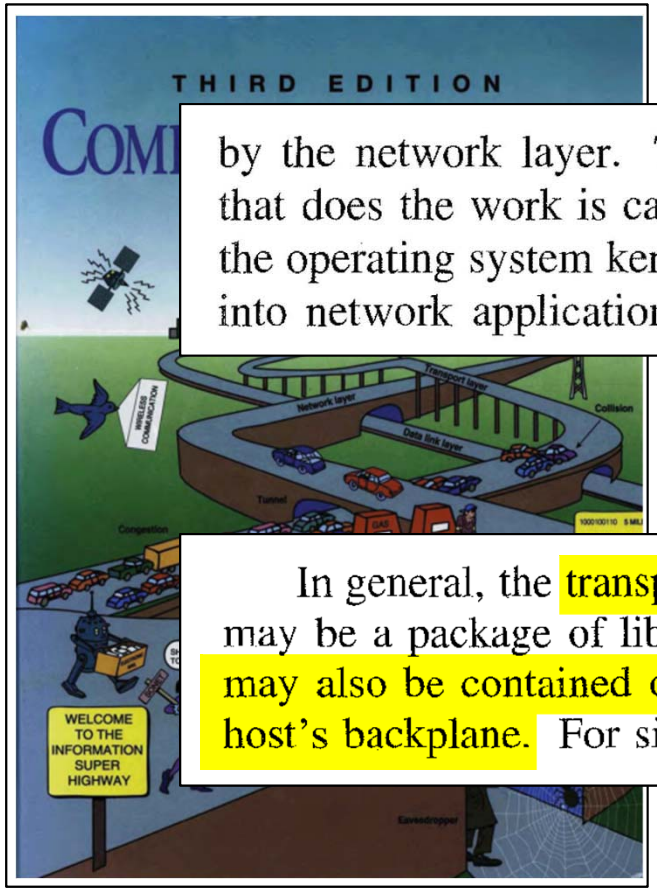
A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64K bytes (in practice, usually about 1500 bytes), and sends each piece as a separate IP datagram. When IP datagrams containing TCP data

Paper 1 (072 Petition) at 43;
Paper 46 (072 Reply) at 15;
Ex. 1003.100 (072 Horst Decl.);
Ex. 1006.540 (Tanenbaum96).

The sending and receiving TCP entities exchange data in the form of segments. A **segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments

Paper 1 (072 Petition) at 43;
Paper 46 (072 Reply) at 15;
Ex. 1003.100 (072 Horst Decl.);
Ex. 1006.543 (Tanenbaum96).

Tanenbaum96: Transport entity may reside on network interface



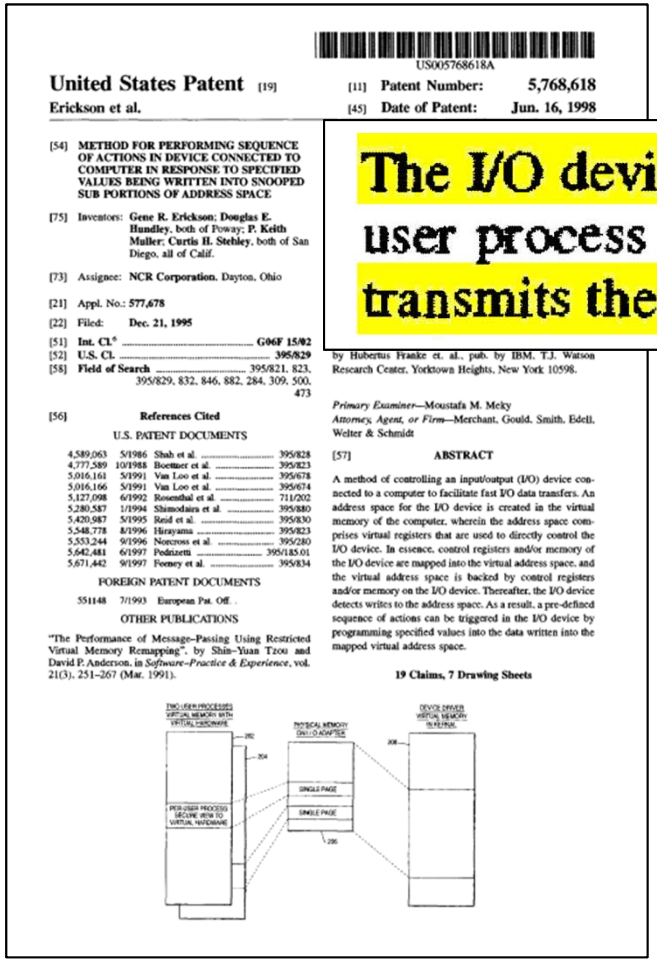
by the network layer. The hardware and/or software within the transport layer that does the work is called the **transport entity**. The transport entity can be in the operating system kernel, in a separate user process, in a library package bound into network applications, or on the network interface card. In some cases, the

Paper 1 (072 Petition) at 43-44;
Ex. 1003.100 (072 Horst Decl.);
Ex. 1006.498 (Tanenbaum96).

In general, the transport entity may be part of the host's operating system or it may be a package of library routines running within the user's address space. It may also be contained on a coprocessor chip or network board plugged into the host's backplane. For simplicity, our example has been programmed as though it

Paper 46 (072 Reply) at 6;
Ex. 1006.530 (Tanenbaum96).

Erickson teaches that its interface device stores and transmits user data



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson**; **Douglas E. Hundley**, both of Poway; **P. Keith Muller**; **Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Edell, Welter & Schmidt

[57] **ABSTRACT**

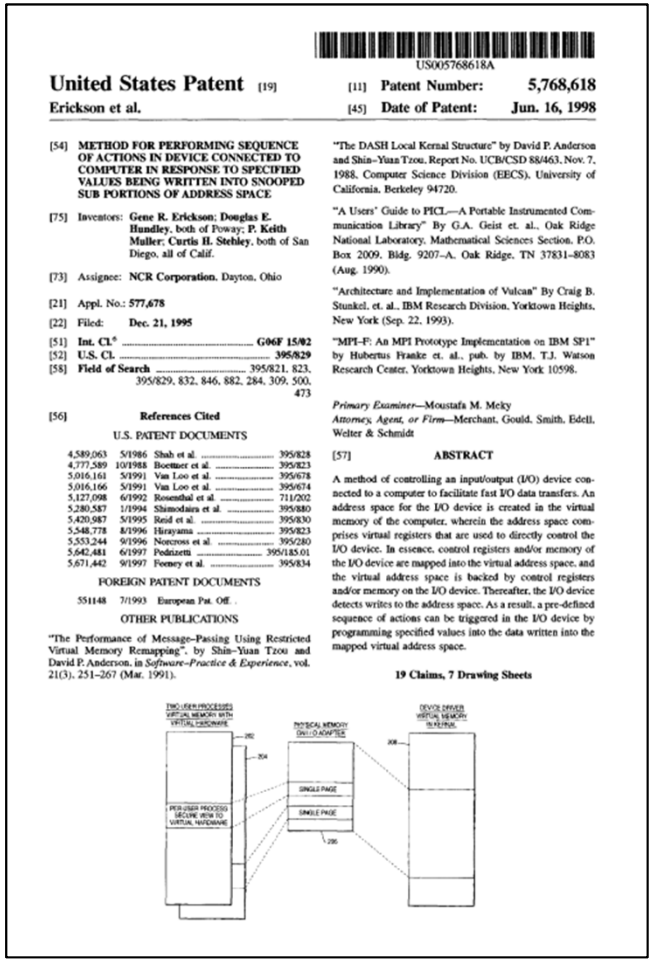
A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

The I/O device adapter stores the user data provided by the user process in the I/O device adapter's memory, and then transmits the completed UDP datagram 702 over the media.

Paper 1 (072 Petition) at 33, 42-43, 44;
Paper 46 (072 Reply) at 15;
Ex. 1003.100-.101 (072 Horst. Decl.);
Ex. 1005 (Erickson) at 7:39-46.

Erickson divides the data and transmits data using adapter



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson**; **Douglas E. Hundley**, both of Poway; **P. Keith Muller**; **Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

"The DASH Local Kernal Structure" by David P. Anderson and Shin-Yuan Tzou, Report No. UCB/CSD 88/463, Nov. 7, 1988, Computer Science Division (EECS), University of California, Berkeley 94720.

"A Users' Guide to PICL—A Portable Instrumented Communication Library" By G.A. Geist et. al., Oak Ridge National Laboratory, Mathematical Sciences Section, P.O. Box 2009, Bldg. 9207-A, Oak Ridge, TN 37831-8083 (Aug. 1990).

"Architecture and Implementation of Vulcan" By Craig B. Stunkel, et. al., IBM Research Division, Yorktown Heights, New York (Sep. 22, 1993).

"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Edell, Welter & Schmidt

[57] **ABSTRACT**

A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

```
udpscript (void *USERDATA_ADDRESS,
            int         USERDATA_LENGTH,
                          template_t *template)
{
char *physaddress;
template->IP.TotalLength = sizeof (IPHeader) +
            sizeof(UDPHeader) + USERDATA_LENGTH;
template->IP.DatagramID = nextid() ;
ipchecksum (template) ;
template->UDPLength = sizeof (UDPHeader)
            + USERDATA_LENGTH;
physaddress = vtophys (USERDATA_ADDRESS,
            USERDATA_LENGTH) ;
udpchecksum (physaddress, USERDATA_LENGTH, template) ;
}
```

for transmission over the media. Instead, the adapter would most likely retrieve the needed user data from the user process' virtual address space using direct memory access (DMA) into the main memory over the bus and retrieving the user data into some portion of the adapter's memory, where it could be referenced more efficiently. The program-

Paper 1 (072 Petition) at 44-46;
Ex. 1003.100-.101 (072 Horst Decl.); Ex. 1005 (Erickson) at 7:51-64, 8:30-35.

Erickson's single page embodiment can support TCP

- POSA would understand a typical page of virtual address space (4K bytes) would be greater than a typical MSS segment (1500 bytes)

> The second script would transfer (via DMA) and transmit one maximum-segment-size (MSS) segment of data "(in practice, usually about 1500 bytes)" (Ex.1006, Tanenbaum at .540) at a time from the block identified by the user data address pointer and length passed to the script, until all of the data from the block was sent. A POSA would understand that typical page sizes in 1996 were larger than 1500 bytes (e.g. 4K pages were common).

Paper 46 (072 Reply) at 16-17;
Ex. 1003.103 (072 Horst Decl.).

I/O traffic would not be a factor in successfully implementing a TCP script

46. In Erickson, or other network implementations, the bulk of the I/O bus traffic is to transfer data (not control information) between the host memory and the adapter. In Erickson's original UDP scripts as well as all three proposed TCP scripts, additional data movement across the I/O bus is not necessary. Erickson Ex. 1005 at 8:27-37. In two of the proposed TCP scripts, the data transfer is performed by multiple segment-sized DMA operations, while in the third script, the data is transferred with one large DMA operation. Ex. 1003, ¶ 143, Ex.1003.102-.103. The total amount of data transferred would be the same in all cases. The differences in control traffic over the I/O bus for various options is small and certainly not a factor in whether or not TCP could have been successfully implemented.

Paper 46 (072 Reply) at 16-17;
Ex. 1223 (072 Horst Reply Decl.) ¶ 46.

072 Patent: Disputes

2. The combination of Erickson and Tanenbaum96 discloses the limitations of claims 1-7 of the 036 Patent

   a) The prior art discloses "dividing, by the interface device, the data into segments" (all claims)

   b) <u>The prior art discloses "transferring status information for the context to the interface device during the same operation as transferring protocol header information to the interface device" (claim 2)</u>

   c) The prior art discloses "receiving, by the interface device, receive packets that correspond to the [context/protocol information], and updating the [context/status information] by the interface device to account for the receive packets" (claims 7, 14, and 21)

“transferring status information ... during the same operation as ... protocol header information”

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,673,072 B2**
(45) **Date of Patent: Mar. 2, 2010**

(54) **FAST-PATH APPARATUS FOR TRANSMITTING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 435 days.

(21) Appl. No.: **11/821,820**

(22) Filed: **Jun. 25, 2007**

(65) **Prior Publication Data**

US 2008/0126553 A1 May 29, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/260,112, filed on Sep. 27, 2002, now Pat. No. 7,237,036, which is a continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a continuation-in-part of application No. 10/023,240, filed on Dec. 17, 2001, now Pat. No. 6,965,941, which is a continuation-in-part of application No. 09/464,283, filed on Dec. 15, 1999, now Pat. No. 6,427,173, which is a continuation-in-part of application No. 09/439,603, filed on Nov. 12, 1999, now Pat. No. 6,247,060, which is a continuation-in-part of application No. 09/067,544, filed on Apr. 27, 1998, now Pat. No. 6,226,680, said application No. 10/260,112 is a continuation-in-part of application No. 09/384,792, filed on Aug. 27, 1999, now Pat. No. 6,434,620, which is a continuation-in-part of application No. 09/141,713, filed on Aug. 28, 1998, now Pat. No. 6,389,479, said application No. 10/260,112 is a continuation-in-

14, 1997, provisional application No. 60/098,296, filed on Aug. 27, 1998.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/230; 709/233; 709/234; 709/232; 709/239; 370/235; 370/468; 370/237; 370/230; 370/233

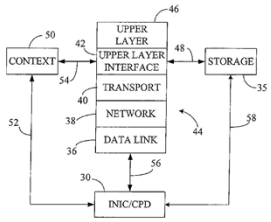
(58) **Field of Classification Search** 709/245, 709/230–234, 239, 250; 370/235, 468, 237, 370/230, 233, 234

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8



2. The method of claim **1**, further comprising transferring status information for the context to the interface device during the same operation as transferring protocol header information to the interface device.

Ex. 1001 (072 Patent), Claim 2.

072 patent: “Status information” can be “basic frame header” information

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,673,072 B2**
(45) **Date of Patent: Mar. 2, 2010**

(54) **FAST-PATH APPARATUS FOR TRANSMITTING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 435 days.

(21) Appl. No.: **11/821,820**

(22) Filed: **Jun. 25, 2007**

(65) **Prior Publication Data**

US 2008/0126553 A1 May 29, 2008

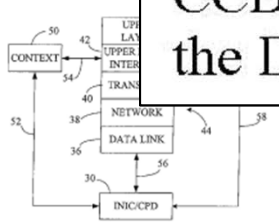
Related U.S. Application Data

(63) Continuation of application No. 10/260,112, filed on Sep. 27, 2002, now Pat. No. 7,237,036, which is a continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a continuation-in-part of application No. 10/023,240, filed on Dec. 17, 2001, now Pat. No. 6,965,941, which is a continuation-in-part of application No. 09/464,283, filed on Dec. 15, 1999, now Pat. No. 6,427,173, which is a continuation-in-part of application No. 09/439,603, filed on Nov. 12, 1999, now Pat. No. 6,247,060, which is a continuation-in-part of application No. 09/067,544, filed on Apr. 27, 1998, now Pat. No. 6,226,680, said application No. 10/260,112 is a continuation-in-part of application No. 09/384,792, filed on Aug. 27, 1999, now Pat. No. 6,434,620, which is a continuation-in-part of application No. 09/141,713, filed on Aug. 28, 1998, now Pat. No. 6,389,479, 6,807,581, and a continuation-in-part of application No. 09/514,425, filed on Feb. 28, 2000, now Pat. No. 6,427,171, and a continuation-in-part of application No. 09/416,925, filed on Oct. 13, 1999, now Pat. No. 6,470,415.

(60) Provisional application No. 60/061,809, filed on Oct. 14, 1997, provisional application No. 60/098,296, filed on Aug. 27, 1998.

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/230; 709/233; 709/234; 709/232; 709/239; 370/235; 370/468;



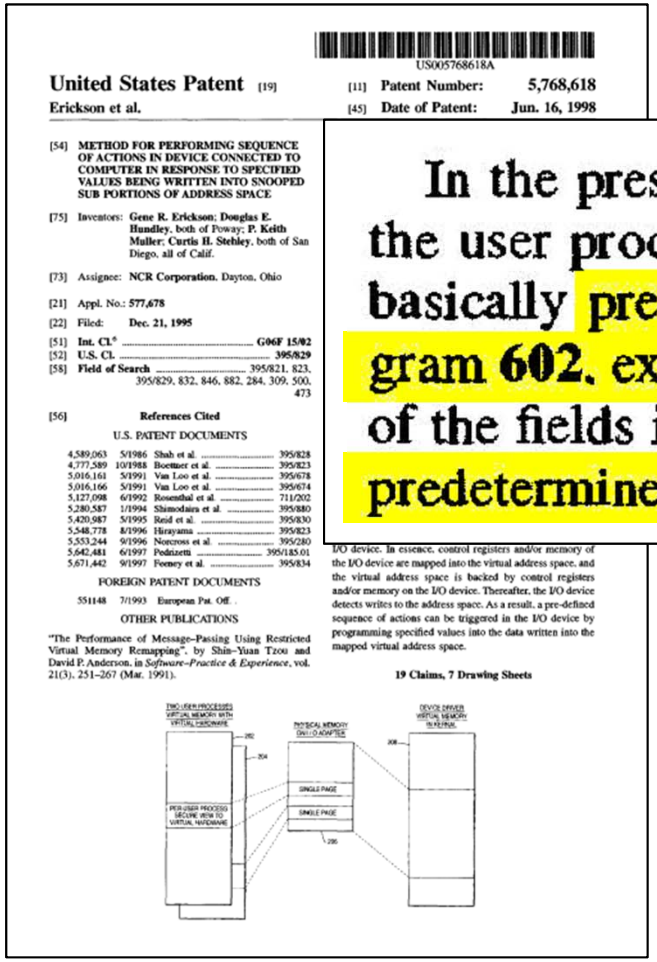
3. The method of claim **1**, wherein creating headers for the segments includes adding status information to the template header.

Paper 1 (072 Petition) 51;
Ex. 1001 (072 Patent) at claim 3.

the payload data. It is quicker and simpler to keep a basic frame header (i.e., a template header) permanently in the CCB and DMA this directly from the SRAM CCB buffer into the DRAM buffer each time. Thus the payload checksum is

Paper 1 (072 Petition) 51;
Ex. 1001 (072 Patent) at 32:56-59.

Erickson teaches “almost everything” about datagram is “pre-negotiated”



US005768618A

United States Patent [19]
Erickson et al.

[11] Patent Number: 5,768,618
[45] Date of Patent: Jun. 16, 1998

[54] METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE

[75] Inventors: Gene R. Erickson; Douglas E. Hundley, both of Poway; P. Keith Muller; Curtis H. Stehley, both of San Diego, all of Calif.

[73] Assignee: NCR Corporation, Dayton, Ohio

[21] Appl. No.: 577,678

[22] Filed: Dec. 21, 1995

[51] Int. Cl.[6] G06F 15/02
[52] U.S. Cl. 395/829
[58] Field of Search 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] References Cited

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

“The Performance of Message-Passing Using Restricted Virtual Memory Remapping”, by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

In the present application, the access privileges given to the user processes are very narrow. Each user process has basically pre-negotiated almost everything about the datagram **602**, except the actual user data **610**. This means most of the fields in the three header areas **604**, **606**, and **608** are predetermined.

Paper 1 (072 Petition) at 53;
Ex. 1005 (Erickson) at 6:57-6:62.

Erickson's datagram template is a basic frame header including status information

FIG. 7

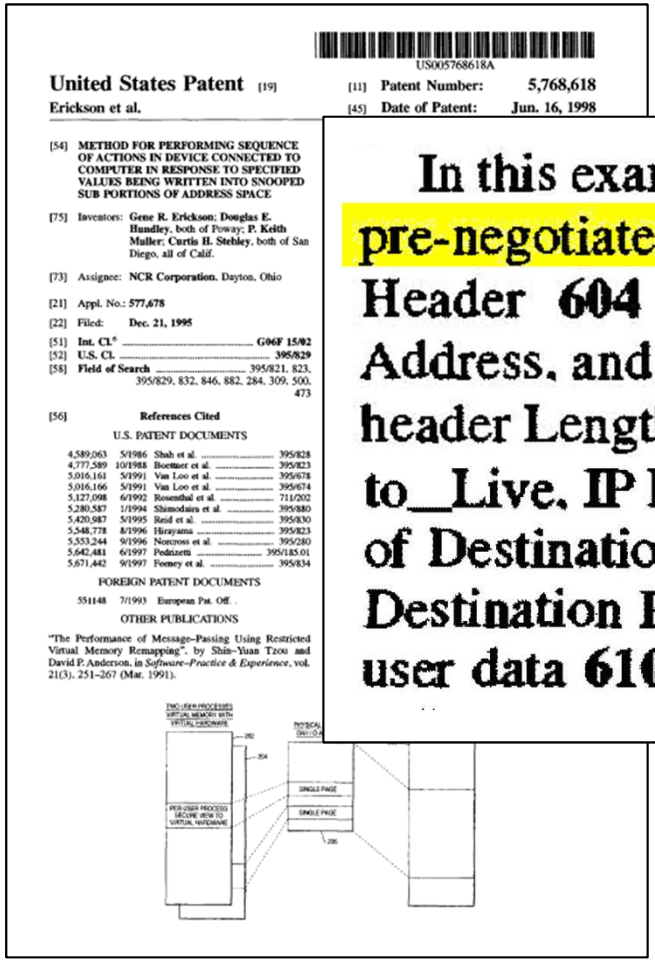
702

Hex	Dec			
0	0	Ethernet Header (14 bytes) 704	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Addres
7	7		08	(6 bytes)
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 =
d	13		00	
e	14	IP Header (20 bytes) 706	45	Version = 4, IP Header l
f	15		00	Service Type
10	16			Total Length
11	17			
12	18			Datagram Id
13	19			
14	20		40	Flag 0x4 DO_NOT_FRA
15	21		00	Fragment Offset = 0x00
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDF
18	24			IP Header Checksum
19	25			
1a	26		80	IP Address of Source =
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destinatio
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 708	00	Source Port = 0x0007 (e
23	35		07	
24	36		30	Destination Port = 0x301
25	37		18	
26	38			UDP Length
27	39			
28	40			UDP Checksum
29	41			

The script that executes the above function provides the USERDATA_ADDRESS and USERDATA_LENGTH which the user process programmed into the adapter's memory. This information quite likely varies from datagram **602** to datagram **602**. The script is also passed the appropriate datagram **702** template based on the specific software register (**508** in FIG. **5** or **316** in FIG. **3**). There are different scripts for different types of datagrams **702** (e.g., UDP or TCP). Also, the script would most likely make a copy of the datagram **702** template (not shown here), so that multiple datagrams **602** for the same user could be simultaneously in transit.

Paper 1 (072 Petition) at 53;
Ex.1003.111 (072 Horst Decl.);
Ex. 1005 (Erickson) at 8:2-4.

“Same operation” because protocol header includes status information



US005768618A

United States Patent [19]

Erickson et al.

[11] Patent Number: 5,768,618

[45] Date of Patent: Jun. 16, 1998

[54] METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE

[75] Inventors: Gene R. Erickson; Douglas E. Hundley, both of Poway; P. Keith Muller; Curtis H. Stehley, both of San Diego, all of Calif.

[73] Assignee: NCR Corporation, Dayton, Ohio

[21] Appl. No.: 577,678

[22] Filed: Dec. 21, 1995

[51] Int. Cl.[6] G06F 15/02

[52] U.S. Cl. 395/829

[58] Field of Search 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] References Cited

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251–267 (Mar. 1991).

In this example, the user process and the device driver has pre-negotiated the following fields from FIG. 6: (1) Ethernet Header **604** (Target Ethernet Address, Source Ethernet Address, and Protocol Type); (2) IP Header **606** (Version, IP header Length, Service Type, Flag, Fragment Offset, Time_to_Live, IP Protocol, IP Address of Source, and IP Address of Destination); and (3) UDP Header **608** (Source Port and Destination Port). Only the shaded fields in FIG. 6, and the user data **610**, need to be changed on a per-datagram basis.

Paper 1 (072 Petition) at 40, 53;
Ex. 1003.074, .112 (072 Horst Decl.);
Ex.1005 (Erickson) at 6:63-7:4.

072 Patent: Disputes

2. The combination of Erickson and Tanenbaum96 discloses the limitations of claims 1-7 of the 036 Patent

   a) The prior art discloses “dividing, by the interface device, the data into segments” (all claims)

   b) The prior art discloses “transferring status information for the context to the interface device during the same operation as transferring protocol header information to the interface device” (claim 2)

   c) <u>The prior art discloses “receiving, by the interface device, receive packets that correspond to the [context/protocol information], and updating the [context/status information] by the interface device to account for the receive packets” (claims 7, 14, and 21)</u>

“Receiving .... and updating ... by the interface device”

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,673,072 B2**

(54) **FAST-PATH APPARATUS FOR TRANSMITTING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of th patent is extended or adjusted under 3 U.S.C. 154(b) by 435 days.

(21) Appl. No.: **11/821,820**

(22) Filed: **Jun. 25, 2007**

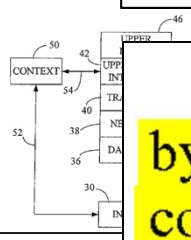
(65) **Prior Publication Data**

US 2008/0126553 A1 May 29, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/260,112, filed Sep. 27, 2002, now Pat. No. 7,237,036, which is continuation of application No. 10/092,967, filed Mar. 6, 2002, now Pat. No. 6,591,302, which is continuation-in-part of application No. 10/023,24 filed on Dec. 17, 2001, now Pat. No. 6,965,941, whi is a continuation-in-part of application No. 09/46 283, filed on Dec. 15, 1999, now Pat. No. 6,427,17 which is a continuation-in-part of application N 09/439,603, filed on Nov. 12, 1999, now Pat. N 6,247,060, which is a continuation-in-part of applic tion No. 09/067,544, filed on Apr. 27, 1998, now P No. 6,226,680, said application No. 10/260,112 is continuation-in-part of application No. 09/384,79 filed on Aug. 27, 1999, now Pat. No. 6,434,620, whi is a continuation-in-part of application No. 09/14 713, filed on Aug. 28, 1998, now Pat. No. 6,389,47

6,427,171, and a continuation-in-part of application No. 09/416,925, filed on Oct. 13, 1999, now Pat. No.



7. The method of claim **1**, further comprising receiving, by the interface device, receive packets that correspond to the context, and updating the context by the interface device to account for the receive packets.

14. The method of claim **9**, further comprising receiving, by the interface device, receive packets that correspond to the protocol information, and updating the status information by the interface device to account for the receive packets.

21. The method of claim **15**, further comprising receiving, by the interface device, receive packets that correspond to the context, and updating the status information by the interface device to account for the receive packets.

Ex. 1001 (072 Patent), Claims 7, 14, 21.

PO ignores the teaching of the combination

- Patent Owner argues about each reference separately:

> 70-71, 76; Ex. 2026, ¶ 105.) Accordingly, Erickson does not disclose receiving packets and updating the corresponding context or status information by the interface device. Tanenbaum discloses conventional packet processing including packet reception that, as described above in § VIII.A, is performed entirely on the host. (Ex. 1006.584-589; Ex. 2026, ¶ 106.) Thus, at least the limitation of

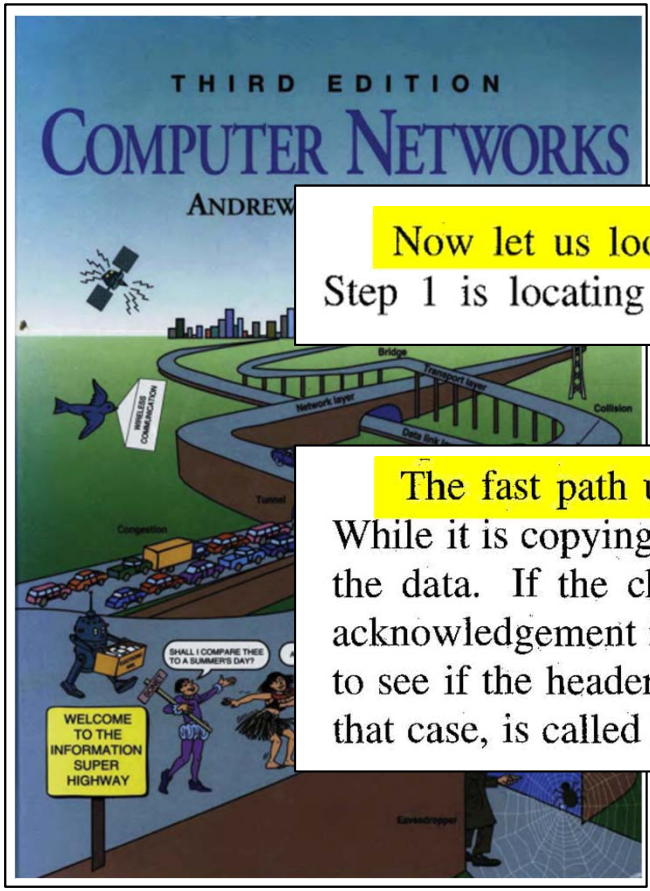
Paper 34 (072 Response) at 34.

- But Petitioner relies on Erickson in view of Tanenbaum96:

> *Id.* at .498 (underlining added, bold in original). Accordingly, it would be obvious to one consulting Tanenbaum96 to implement Erickson's fast path TCP protocol processing for the I/O device adapter to perform *"receiving, by the interface device, receive packets that correspond to the context, and updating the context by the interface device to account for the receive packets."*

Paper 1 (072 Petition) at 62.

Obvious to use Tanenbaum96's fast-path connection records with Erickson's adapter



Now let us look at fast path processing on the receiving side of Fig. 6-49. Step 1 is locating the connection record for the incoming TPDU. For ATM,

The fast path updates the connection record and copies the data to the user. While it is copying, it also computes the checksum, eliminating an extra pass over the data. If the checksum is correct, the connection record is updated and an acknowledgement is sent back. The general scheme of first making a quick check to see if the header is what is expected, and having a special procedure to handle that case, is called **header prediction**. Many TCP implementations use it. When

Paper 1 (072 Petition) at 61-62, 70-71;
Ex. 1003.110, .121-.122 (072 Horst Decl.);
Ex. 1006.584-.585 (Tanenbaum96).

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

a) **<u>Patent Owner has improperly expanded the scope of claims 22-29</u>**

b) Patent Owner does not show adequate written description support

c) Substitute claims 22-29 are indefinite

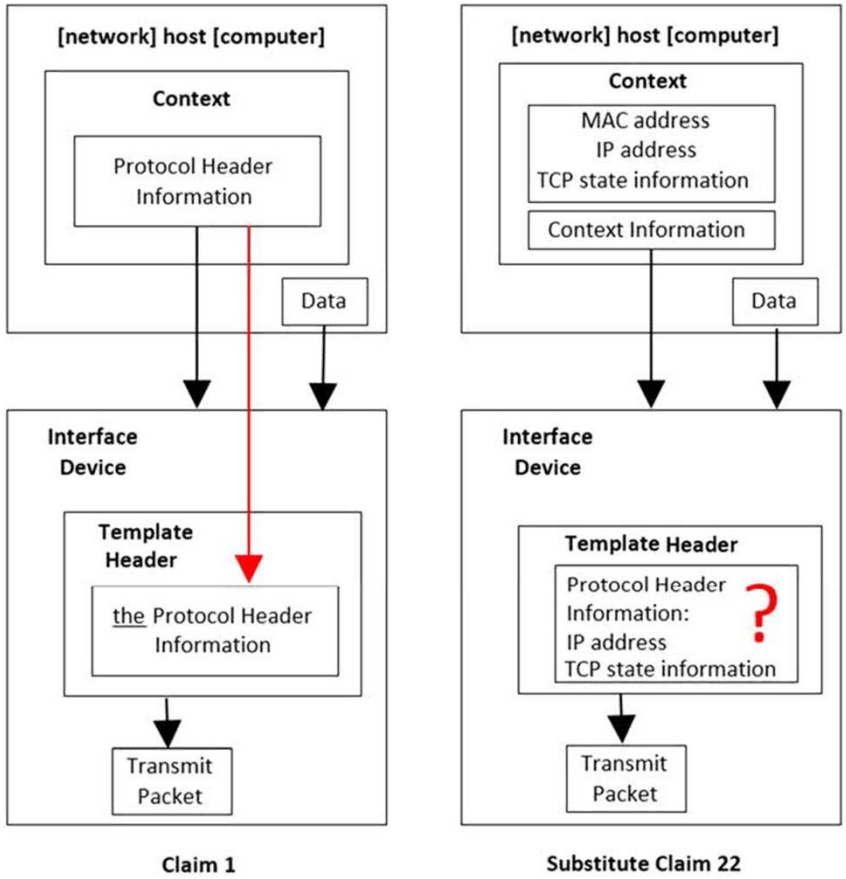
d) Substitute claims are obvious

“Protocol header information” untied to any other information in substitute claim 22

Original Claim 1	Substitute Claim 22
[1] A method comprising:	[22] A method comprising:
[1.1] establishing, at a host computer, a transport layer connection, including creating a context that includes protocol header information for the connection;	[22.1] establishing, at a host computer, a transport layer connection, including creating a context that includes **a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information** ~~protocol header information~~ for the connection;
[1.2] transferring the protocol header information to an interface device;	[22.2] transferring the **context** ~~protocol header~~ information to an interface device;
[1.3] transferring data from the network host to the interface device, after transferring the protocol header information to the interface device;	[22.3] transferring data from the network host to the interface device, after transferring the **context** ~~protocol header~~ information to the interface device;
[1.4] dividing, by the interface device, the data into segments;	[22.4] dividing, by the interface device, the data into segments;
[1.5] creating headers for the segments, by the interface device, from a template header containing the protocol header information; and	[22.5] creating headers for the segments, by the interface device, from a template header containing ~~the~~ protocol header information **including IP address and TCP state information**; and
[1.6] prepending the headers to the segments to form transmit packets.	[22.6] prepending the headers to the segments to form transmit packets.

Paper 54 (072 Sur-Reply for Opp. to Motion to Amend) at 3.

“Protocol header information” untied to “context” would infringe claim 22, but not claim 1



Paper 40 (072 Opp. to Motion to Amend) at 7.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

   a) Patent Owner has improperly expanded the scope of claims 22-29

   b) <u>Patent Owner does not show adequate written description support</u>

   c) Substitute claims 22-29 are indefinite

   d) Substitute claims are obvious

PO must supply written description support after *Aqua Products*

United States Patent and Trademark Office
Office of the Chief Administrative Patent Judge

MEMORANDUM

TO: Patent Trial and Appeal Board

FROM: David P. Ruschke
Chief Administrative Patent Judge
David.Ruschke@uspto.gov

DATE: November 21, 2017

RE: **Guidance on Motions to Amend in view of *Aqua Products***

Beyond that change, generally speaking, practice and procedure before the Board will not change. For example, a patent owner still must meet the requirements for a motion to amend under 37 C.F.R. § 42.121 or § 42.221, as applicable. That is, a motion to amend must set forth written description support and support for the benefit of a filing date in relation to each substitute claim, and respond to grounds of unpatentability involved in the trial. Likewise, under 37 C.F.R. § 42.11, all parties have a duty of

Paper 54 (072 Sur-Reply for Opp. to Motion to Amend) at 5;
Nov. 21, 2017 USPTO Memo

PO identifies same 10 pages and 12 figures for every independent claim limitation

Claims	Exemplary Support in the '820 Application
Proposed Claim 22	
~~1~~22. A method comprising:	*See* below.
establishing, at a host computer, a transport layer connection, including creating a context that includes a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information ~~protocol header information~~ for the connection;	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
transferring the context ~~protocol header~~ information to an interface device;	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
transferring data from the network host to the interface device, after transferring the context ~~protocol header~~ information to the interface device;	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
dividing, by the interface device, the data into segments;	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
creating headers for the segments, by the interface device, from a template header containing ~~the~~ protocol header information including IP address and TCP state information; and	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
prepending the headers to the segments to form transmit packets.	*See*, *e.g.*, Ex. 2024 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.

Same written description support as 036 Patent

Paper 40 (072 Opp. to Motion to Amend) at 4-5; Paper 62 (072 Corrected Exhibits for Motion to Amend), Appx. A.

Too late to provide written description support in Reply

- PO provides alleged "exemplary" written description support **for the first time** in its Reply

VII. THE PROPOSED AMENDMENTS ARE SUPPORTED BY THE WRITTEN DESCRIPTION

75. It is my opinion that the proposed amendments are supported by the written description, along with the Application (No. 11/821,820) and Provisional Application (No. 60/061,809).

Paper 47 (072 Reply ISO Motion to Amend) at 6;
Ex. 2305 (Almeroth Decl. ISO Reply) at 25.

Written description support provided by PO is insufficient

- Patent Owner cites to written description support not included in its original motion

- Patent Owner has not identified any written description support for:
  - "Transferring the context information to an interface device"
  - Creating a "template header" from **<u>any</u>** "protocol header information"

Paper 54 (072 Sur-Reply for Motion to Amend) at 7-8.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

a) Patent Owner has improperly expanded the scope of claims 22-29

b) Patent Owner does not show adequate written description support

c) <u>Substitute claims 22-29 are indefinite</u>

d) Substitute claims are obvious

Claim 22 is indefinite

(22.1) establishing, at a host computer, a transport layer connection, including creating **a context** that includes a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information for the connection;

(22.2) transferring **the context information** to an interface device;

(22.3) transferring data from the network host to the interface device, after transferring **the context information** to the interface device;

Paper 40 (072 Opp. to Motion to Amend) at 8-10;
Ex. 1210 (072 Horst Decl. ISO Opp. to Motion to Amend) ¶ 18.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

d) <u>Substitute claims are obvious</u>

i. **<u>Prior art discloses "creating a context that includes a MAC layer address, an IP address, and TCP state information for the connection" (limitation 22.1)</u>**

ii. Prior art discloses "transferring the context information to an interface device" (limitation 22.2)

iii. Prior art discloses "transferring data ... after transferring the context information to the interface device" (limitation 22.3)

iv. Prior art discloses "creating headers for the segments, by the interface device, from a template header" containing "TCP state information" (limitation 22.5)

Erickson: “Pre-negotiated” header that includes “almost everything” for UDP

FIG. 6

602

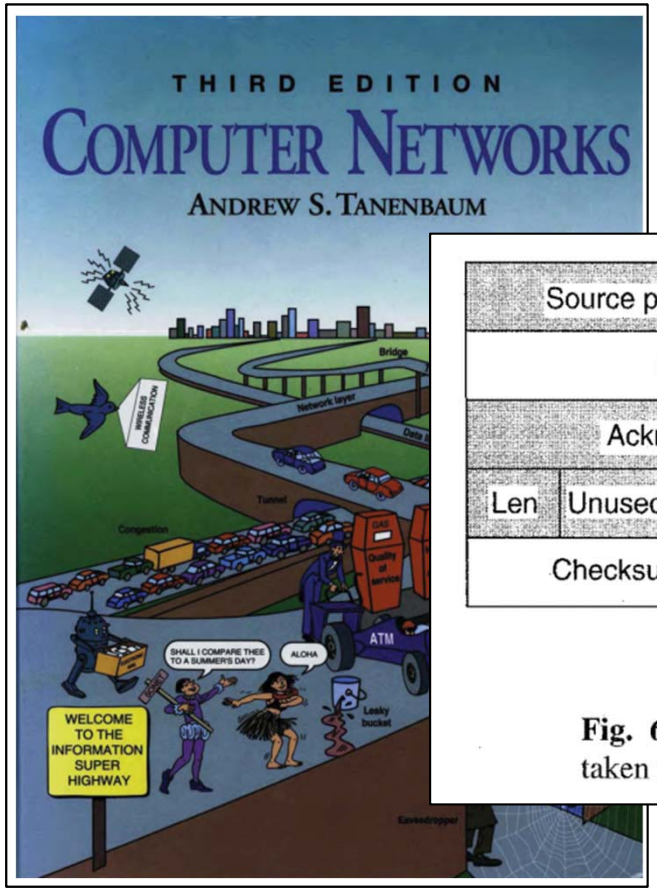
Hex	Dec			
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Address (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 = IP)
d	13		00	
e	14	IP Header (20 bytes) 606	45	Version = 4, IP Header Len (Words) = 5
f	15		00	Service Type
10	16		00	Total Length = 0x001d (29 bytes: 20-byte IP Header plus 8-byte UDP header plus 1-byte user data)
11	17		1d	
12	18		e0	Datagram Id = 0xe0a1
13	19		a1	
14	20		40	Flag 0x4 DO_NOT_FRAGMENT
15	21		00	Fragment Offset = 0x000
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP)
18	24		da	IP Header Checksum = 0xda1b
19	25		1b	
1a	26		80	IP Address of Source = 128.1.192.7
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destination = 128.1.192.8
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 608	00	Source Port = 0x0007 (echo datagram)
23	35		07	
24	36		30	Destination Port = 0x3018
25	37		18	
26	38		00	UDP Length = 0x0009 (8-byte UDP Header plus 1-byte user data)
27	39		09	
28	40		0c	UDP Checksum = 0x0cf8
29	41		f8	
2a	42	User Data (Variable) 610	67	1 byte user datagram = "g"

In the present application, the access privileges given to the user processes are very narrow. Each user process has basically pre-negotiated almost everything about the datagram **602**, except the actual user data **610**. This means most of the fields in the three header areas **604**, **606**, and **608** are predetermined.

In this example, the user process and the device driver has pre-negotiated the following fields from FIG. **6**: (1) **Ethernet Header 604** (Target Ethernet Address, Source Ethernet Address, and Protocol Type); (2) **IP Header 606** (Version, IP header Length, Service Type, Flag, Fragment Offset, Time_to_Live, IP Protocol, IP Address of Source, and IP Address of Destination); and (3) **UDP Header 608** (Source Port and Destination Port). Only the shaded fields in FIG. **6**, and the user data **610**, need to be changed on a per-datagram basis.

Paper 40 (072 Opp. to Motion to Amend) at 11-12;
Ex. 1005 (Erickson) at 6:58-7:3, Fig. 6.

POSA would have replaced UDP header with TCP header



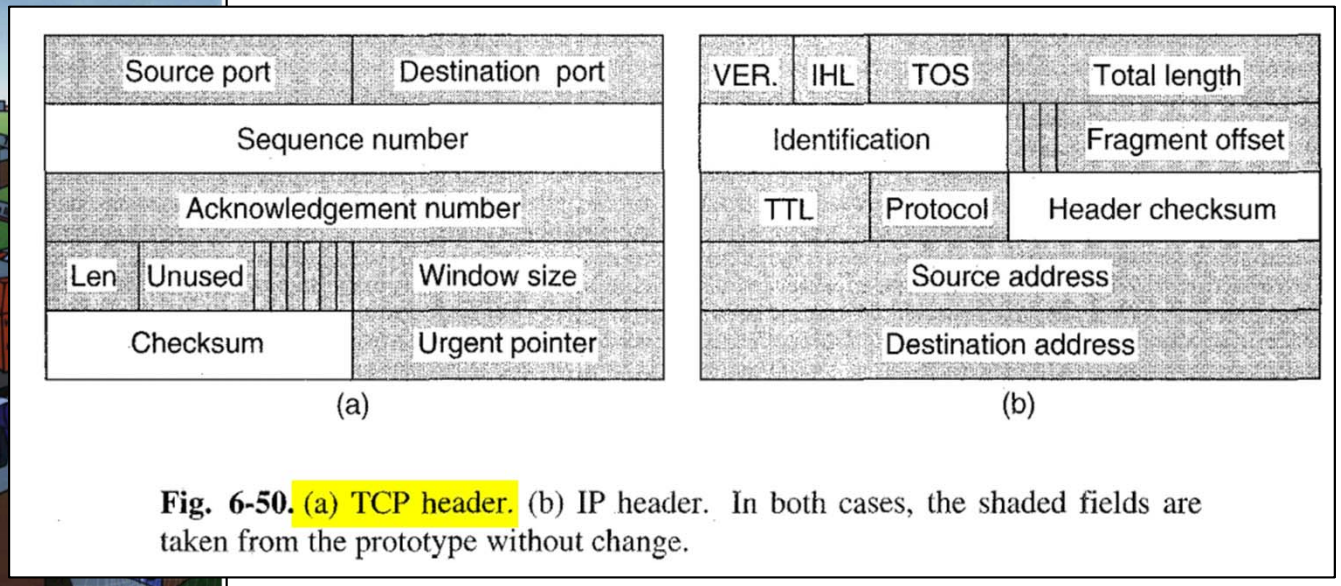


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 40 (072 Opp. to Motion to Amend) at 12;
Ex. 1006 (Tanenbaum96) at .584, Fig. 6-50.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

d) <u>Substitute claims are obvious</u>

i. Prior art discloses "creating a context that includes a MAC layer address, an IP address, and TCP state information for the connection" (limitation 22.1)

ii. <u>Prior art discloses "transferring the context information to an interface device" (limitation 22.2)</u>

iii. Prior art discloses "transferring data ... after transferring the context information to the interface device" (limitation 22.3)

iv. Prior art discloses "creating headers for the segments, by the interface device, from a template header" containing "TCP state information" (limitation 22.5)

Erickson discloses transferring "datagram template" to I/O adapter

FIG. 7

702

Hex	Dec			
0	0	Ethernet Header (14 bytes) 704	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Addres (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 =
d	13		00	
e	14	IP Header (20 bytes) 706	45	Version = 4, IP Header l
f	15		00	Service Type
10	16			Total Length
11	17			
12	18			Datagram Id
13	19			
14	20		40	Flag 0x4 DO_NOT_FRA
15	21		00	Fragment Offset = 0x00
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP
18	24			IP Header Checksum
19	25			
1a	26		80	IP Address of Source =
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destinatio
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 708	00	Source Port = 0x0007 (e
23	35		07	
24	36		30	Destination Port = 0x301
25	37		18	
26	38			UDP Length
27	39			
28	40			UDP Checksum
29	41			

The script that executes the above function provides the USERDATA_ADDRESS and USERDATA_LENGTH which the user process programmed into the adapter's memory. This information quite likely varies from datagram **602** to datagram **602**. The script is also passed the appropriate datagram **702** template based on the specific software register (**508** in FIG. **5** or **316** in FIG. **3**). There are different scripts for different types of datagrams **702** (e.g., UDP or TCP). Also, the script would most likely make a copy of the datagram **702** template (not shown here), so that multiple datagrams **602** for the same user could be simultaneously in transit.

Paper 1 (072 Petition) at 42;
Paper 40 (072 Opp. to Motion to Amend) at 13.
Ex. 1005 (Erickson) at 7:65-8:9.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

d) <u>Substitute claims are obvious</u>

i. Prior art discloses "creating a context that includes a MAC layer address, an IP address, and TCP state information for the connection" (limitation 22.1)

ii. Prior art discloses "transferring the context information to an interface device" (limitation 22.2)

iii. <u>Prior art discloses "transferring data ... after transferring the context information to the interface device" (limitation 22.3)</u>

iv. Prior art discloses "creating headers for the segments, by the interface device, from a template header" containing "TCP state information" (limitation 22.5)

Data is transferred after the context information

FIG. 7

702

Hex	Dec			
0	0	Ethernet Header (14 bytes) 704	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Addres (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 =
d	13		00	
e	14	IP Header (20 bytes) 706	45	Version = 4, IP Header l
f	15		00	Service Type
10	16			Total Length
11	17			
12	18			Datagram Id
13	19			
14	20		40	Flag 0x4 DO_NOT_FRA
15	21		00	Fragment Offset = 0x00
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP
18	24			IP Header Checksum
19	25			
1a	26		80	IP Address of Source =
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destinatio
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 708	00	Source Port = 0x0007 (echo datagram)
23	35		07	
24	36		30	Destination Port = 0x3018
25	37		18	
26	38			UDP Length
27	39			
28	40			UDP Checksum
29	41			

FIG. 7 is a block diagram illustrating a UDP datagram template **702** (without a user data area) residing in the I/O device adapter's memory. The user process provides the starting address and the length for the user data in its virtual address space, and then "spanks" a GO register to trigger the I/O device adapter's execution of a predetermined script. The I/O device adapter stores the user data provided by the user process in the I/O device adapter's memory, and then transmits the completed UDP datagram **702** over the media.

Paper 1 (072 Petition) at 42-43;
Paper 40 (072 Opp. to Motion to Amend) at 14;
Ex. 1210.024 (072 Horst Decl. ISO Opp. to Motion to Amend);
Ex. 1005 (Erickson) at 7:39-48, Fig. 7.

072 Patent: Disputes

3. Motion to Amend 072 Patent should be denied

d) <u>Substitute claims are obvious</u>

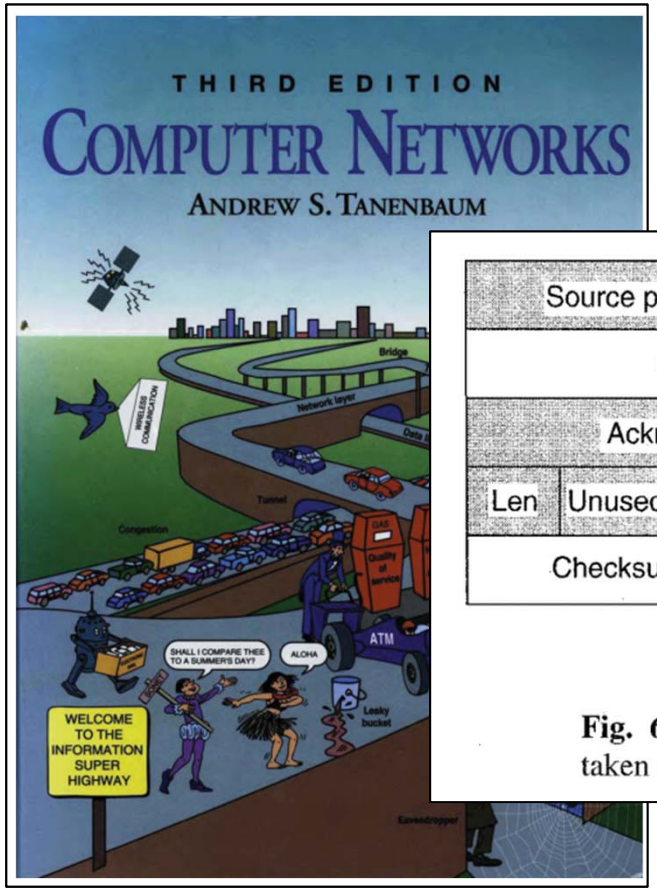
i. Prior art discloses "creating a context that includes a MAC layer address, an IP address, and TCP state information for the connection" (limitation 22.1)

ii. Prior art discloses "transferring the context information to an interface device" (limitation 22.2)

iii. Prior art discloses "transferring data ... after transferring the context information to the interface device" (limitation 22.3)

iv. <u>Prior art discloses "creating headers for the segments, by the interface device, from a template header" containing "TCP state information" (limitation 22.5)</u>

POSA would have replaced UDP header with TCP header



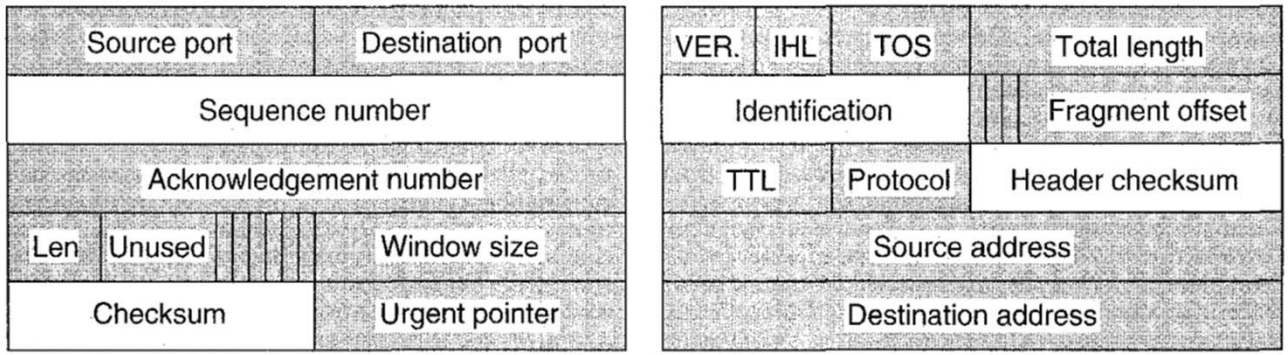


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 40 (072 Opp. to Motion to Amend) at 12;
Ex. 1006 (Tanenbaum96) at .584, Fig. 6-50.

U.S. Patent No. 7,337,241 (241 Patent)

IPR2017-1392 (Intel)
IPR2018-0372 (Dell)
IPR2017-1728 (Cavium)
IPR2018-0328 (Wistron)

All citations herein are to the IPR2017-01405 case unless otherwise noted.

241 Patent: Instituted Grounds

- Erickson in view of Tanenbaum96 and Alteon
  - Claims **1,** 2-8, 18, 22, and 23
- Erickson in view of Tanenbaum96 (Common to 036 and 072 Patents)*
  - Claims **9,** 10-16, **17**, 19-21, and 24

* This combination is discussed on slides 6-53 (regarding 036 and 072 patents)

Ex. 1005 – U.S. Patent No. 5,768,618 ("Erickson")
Ex. 1006 – Tanenbaum, Andrew S., Computer Networks ("Tanenbaum96")
Ex. 1033 – "Gigabit Ethernet Technical Brief ("Alteon")

241 Patent: Disputes

1. A POSA would be motivated to combine
   a. Erickson and Tannenbaum96
   b. Alteon with Erickson and Tannenbaum96
2. The prior art discloses all of the disputed limitations of the 241 Patent
3. Alteon is Prior Art
4. Motion to Amend 241 Patent

241 Patent: Disputes

1. A POSA would be motivated to combine
   a. **<u>Erickson and Tannenbaum96</u>**

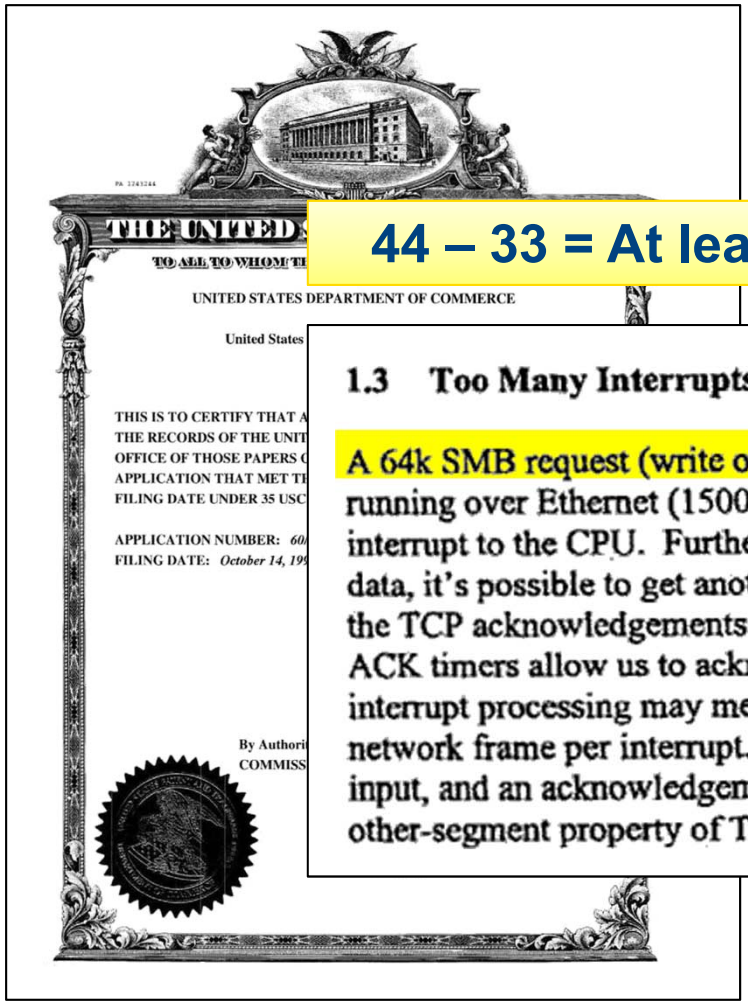
- See 036 Patent, Dispute 1, slides 6-53

241 Patent: Disputes

1. A POSA would be motivated to combine
   a. Erickson and Tannenbaum96
   b. Alteon with Erickson and Tannenbaum96
      i. **<u>A POSA would have been motivated to apply Alteon's interrupt reductions to Erickson and Tanenbaum96</u>**
      ii. Tanenbaum96 does not teach away from the combination (see slides 33-40)
      iii. Erickson and Alteon are compatible

Ex. 1005 – U.S. Patent No. 5,768,618 ("Erickson")
Ex. 1006 – Tanenbaum, Andrew S., Computer Networks ("Tanenbaum96")
Ex. 1033 – "Gigabit Ethernet Technical Brief ("Alteon")

PO has admitted that using fewer than one interrupt per packet was known


THE UNITED

TO ALL TO WHOM T

UNITED STATES DEPARTMENT OF COMMERCE

United States

THIS IS TO CERTIFY THAT A
THE RECORDS OF THE UNIT
OFFICE OF THOSE PAPERS
APPLICATION THAT MET T
FILING DATE UNDER 35 USC

APPLICATION NUMBER: 60
FILING DATE: October 14, 19

By Authori
COMMISS

44 – 33 = At least 11 segments with no interrupts

1.3 Too Many Interrupts

A 64k SMB request (write or read-reply) is typically made up of 44 TCP segments when running over Ethernet (1500 byte MTU). Each of these segments may result in an interrupt to the CPU. Furthermore, since TCP must acknowledge all of this incoming data, it's possible to get another 44 transmit-complete interrupts as a result of sending out the TCP acknowledgements. While this is possible, it is not terribly likely. Delayed ACK timers allow us to acknowledge more than one segment at a time. And delays in interrupt processing may mean that we are able to process more than one incoming network frame per interrupt. Nevertheless, even if we assume 4 incoming frames per input, and an acknowledgement for every 2 segments (as is typical per the ACK-every-other-segment property of TCP), we are still left with 33 interrupts per 64k SMB request.

Paper 45 (241 Reply) at 6;
Ex. 1031.006 (1997 Provisional).

Erickson sought to avoid operating system intervention

169. A chief concern of Erickson is to avoid intervention of the operating system on a "per I/O basis." Ex.1005, Erickson at 3:9-10. Erickson partially addresses this problem by using scripts on the I/O device to process data. Alteon

Ex. 1003.101 (Horst Decl.) at ¶ 169.

Thus, it will be recognized that the present invention increases the efficiency of I/O operations in the following ways:

1. Writing information to and from a user address space without intermediate memory-to-memory copies.
2. Accessing an I/O device simultaneously from multiple user processes in a single node.
3. Eliminating calls to the operating system and the associated context switches on a per I/O basis.

Ex. 1005.010 (Erickson) at 3:1-10.

Alteon shows that sending fewer interrupts was known and desirable

Gigabit Ethernet Technical Brief

Using an intelligent adapter with an onboard RISC-based processor specially designed for embedded application processing, Alteon's Gigabit Ethernet technology not only reduces the number of times data is copied among processing entities, it allows a single interrupt to be issued for multiple data packets—radically altering the ratio of interrupts to packets, and eliminating the scalability problems inherent in older adapter designs.

Alteon Gigabit Ethernet adapters have an interrupt timer that determines when to interrupt a host CPU. This allows a single interrupt to be issued for multiple data packets that are sent into the operating system buffer space. It also allows for "adaptive" interrupts; that is, the NIC can alter the number of interrupts issued per second based on network usage.

Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119

1-408-574-5500

First Edition
September 1996

Paper 4 (241 Petition) at 47-48;
Paper 45 (241 Reply) at 5;
Ex. 1033.022-.023 (Alteon).

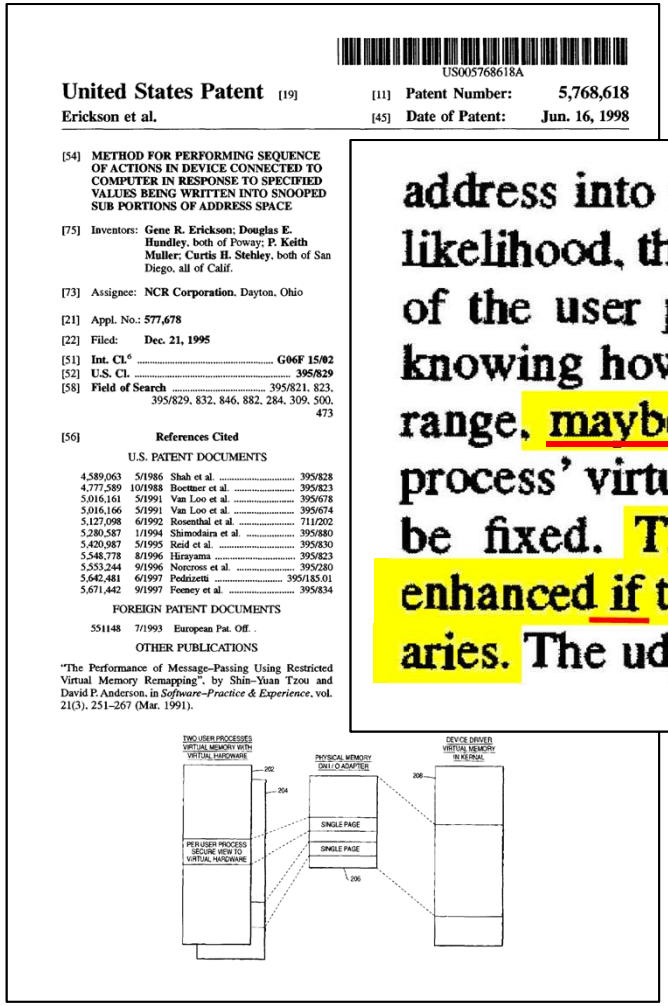
241 Patent: Disputes

1. A POSA would be motivated to combine
   a. Erickson and Tannenbaum96
   b. Alteon with Erickson and Tannenbaum96
      i. A POSA would have been motivated to apply Alteon's interrupt reductions to Erickson and Tanenbaum96
      ii. Tanenbaum96 does not teach away from the combination (see slides 33-40)
      iii. Erickson and Alteon are compatible

241 Patent: Disputes

1. A POSA would be motivated to combine
   a. Erickson and Tannenbaum96
   b. Alteon with Erickson and Tannenbaum96
      i. A POSA would have been motivated to apply Alteon's interrupt reductions to Erickson and Tanenbaum96
      ii. Tanenbaum96 does not teach away from the combination (see slides 33-40)
      iii. Erickson and Alteon are compatible

Erickson is not limited to a single page architecture



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson**; **Douglas E. Hundley**, both of Poway; **P. Keith Muller**; **Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message–Passing Using Restricted Virtual Memory Remapping", by Shin–Yuan Tzou and David P. Anderson, in *Software–Practice & Experience*, vol. 21(3), 251–267 (Mar. 1991).

address into a physical address usable by the adapter. In all likelihood, the adapter would have a very limited knowledge of the user process' virtual address space, probably only knowing how to map virtual-to-physical for a very limited range, maybe as small as a single page. Pages in the user process' virtual address space for such buffers would need to be fixed. The udpscript procedure would need to be enhanced if the user data were allowed to span page boundaries. The udpchecksum( ) procedure generates a checksum

Ex. 1005.012 (Erickson) at 8:16-24;
Ex. 1223.024 (Horst Reply Decl.) at ¶¶ 47-49;
Paper 45 (241 Reply) at 11-12.

A single page is sufficient to hold multiple TCP segments

UNITED STATES PATENT AND TRADEMARK OFFICE

Title: FAS

DECLA
REPLY

Mail Stop
Patent Tri
U.S. Paten
P.O. Box 1450
Alexandria, VA 22313-1450

[1] Cavium, Inc., which filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding. Wistron Corporation, who filed a Petition in Case IPR2018-00328, has been joined as a petitioner in this proceeding.

embodiments in the specification. However, even if Erickson was limited to a single page, TCP segments are often smaller than a page. For example, in Ethernet which is the most common media, TCP segments are typically about 1500 bytes, which is smaller than a typical page size of 4K bytes. Thus, a POSA could have implemented a TCP embodiment without making changes to the Erickson adapter's ability to cross page boundaries.

Ex. 1223.023-.024 (Horst Reply Decl.) at ¶ 46;
Paper 45 (241 Reply) at 11-12.

241 Patent: Disputes

2. The prior art discloses all of the disputed limitations of the 241 Patent

   a) Erickson in view of Tanenbaum96 and Alteon discloses the limitations of claims 1, 2-8, 18, 22, and 23 of the 241 Patent (receive claims)

   b) Erickson in view of Tanenbaum96 discloses the limitations of claims **9**, 10-16, **17**, 19-21, and 21 of the 241 Patent (transmit claims)

241 Patent: Disputes (Receive Claims)

a) Erickson in view of Tanenbaum96 and Alteon discloses the limitations of claims **1**, 2-8, 18, 22, and 23 of the 241 Patent

i. **The prior art discloses validation of network and transport layer headers "without an interrupt dividing the processing" (claim 1)**

ii. The prior art discloses sending the data from each packet to a destination in memory without sending any of the headers (claim **1**)

iii. The prior art discloses processing MAC layer headers without an interrupt (claim 2)

iv. The prior art discloses processing an upper layer header by a second mechanism (claim 3)

v. The prior art discloses sorting the packets by classifying each as having IP and TCP headers (claim 6)

241 Patent: Claim 1

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451
4,991,133	A	2/1991	Davis et al.	

(Continued)

FOREIGN PATENT DOCUMEN

WO WO/98/19412 5/1998

(Continued)

OTHER PUBLICATIONS

Internet pages entitled "Hardware Assisted Proto
(which Eugene Feinberg is working on), 1 page,
1998.

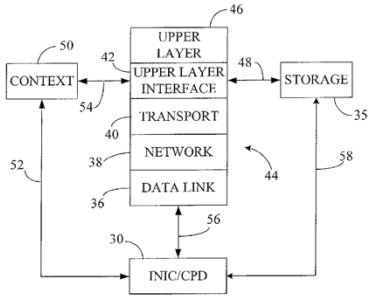
(Continued)

Primary Examiner—David Wiley
(74) *Attorney, Agent, or Firm*—Mark Lauer
Law Group LLP

(57) **ABSTRACT**

A system for protocol processing in a comput
an intelligent network interface card (INIC)
tion processing device (CPD) associated wi
puter. The INIC provides a fast-path that a
processing for most large multi-packet me
accelerating data communication. The INIC
host for those message packets that are chos
ing by host software layers. A communicatio
for a message is defined that allows DMA co
INIC to move data, free of headers, directly
destination or source in the host. The context
INIC as a communication control block (CC
passed back to the host for message processi
The INIC contains specialized hardware ci
much faster at their specific tasks than a g
CPU. A preferred embodiment includes a tri
processors with separate processors devote
receive and management processing, with fu
munication for four fast Ethernet nodes.

24 Claims, 89 Drawing Sheet



1. A method for network communication, the method comprising:

receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;

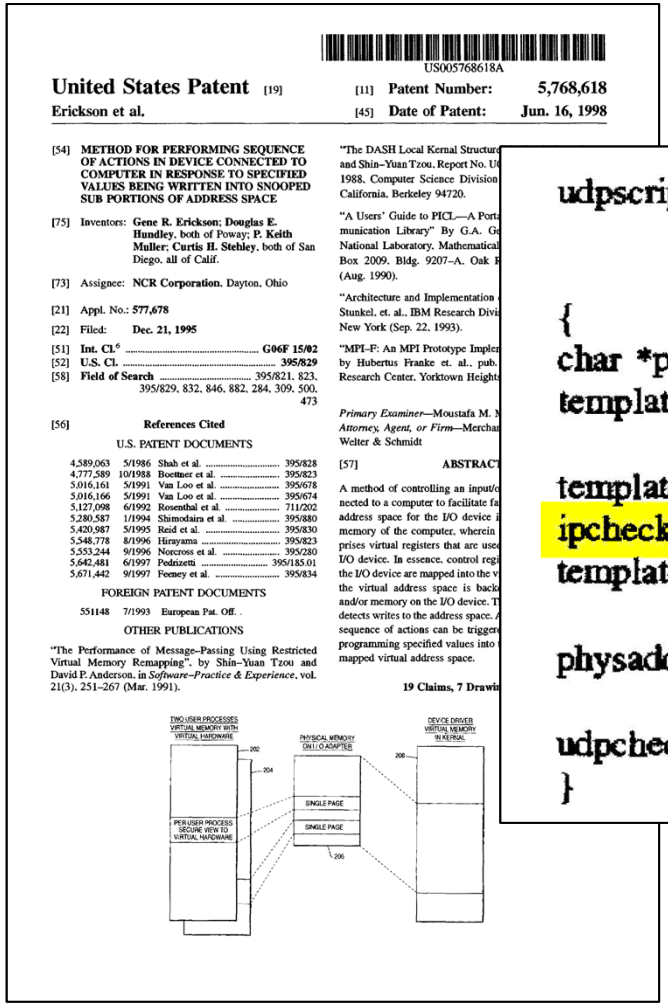
processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;

sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;

sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application without sending any of the media access control layer headers, network layer

Ex. 1001.142 (241 Patent), Claim 1.

The prior art combination teaches header validation on the adapter



US005768618A

United States Patent [19]
Erickson et al.

[11] **Patent Number: 5,768,618**
[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson; Douglas E. Hundley**, both of Poway; **P. Keith Muller; Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**
[52] **U.S. Cl.** **395/829**
[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message–Passing Using Restricted Virtual Memory Remapping", by Shin–Yuan Tzou and David P. Anderson, in *Software–Practice & Experience*, vol. 21(3), 251–267 (Mar. 1991).

"The DASH Local Kernal Structur
and Shin–Yuan Tzou, Report No. U
1988, Computer Science Division
California, Berkeley 94720.

"A Users' Guide to PICL—A Port
munication Library" By G.A. G
National Laboratory, Mathematical
Box 2009, Bldg. 9207–A, Oak
(Aug. 1990).

"Architecture and Implementation
Stunkel, et. al., IBM Research Divi
New York (Sep. 22, 1993).

"MPI–F: An MPI Prototype Impler
by Hubertus Franke et. al., pub.
Research Center, Yorktown Height

Primary Examiner—Moustafa M. N
Attorney, Agent, or Firm—Merchan
Welter & Schmidt

[57] **ABSTRACT**

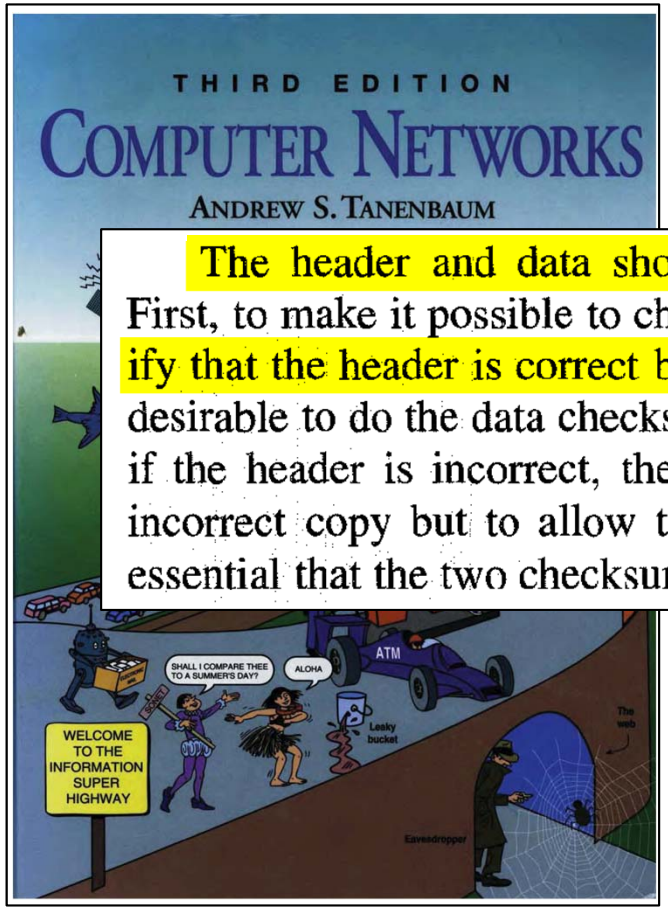
A method of controlling an input/o
nected to a computer to facilitate fa
address space for the I/O device i
memory of the computer, wherein
prises virtual registers that are use
I/O device. In essence, control regi
the I/O device are mapped into the v
the virtual address space is back
and/or memory on the I/O device. T
detects writes to the address space. A
sequence of actions can be trigger
programming specified values into
mapped virtual address space.

19 Claims, 7 Drawi


```
udpscript (void *USERDATA_ADDRESS,
            int          USERDATA_LENGTH,
                              template_t *template)
{
char *physaddress;
template->IP.TotalLength = sizeof (IPHeader) +
            sizeof(UDPHeader) + USERDATA_LENGTH;
template->IP.DatagramID = nextid() ;
ipchecksum (template) ;
template->UDPLength = sizeof (UDPHeader)
            + USERDATA_LENGTH;
physaddress = vtophys (USERDATA_ADDRESS,
            USERDATA_LENGTH) ;
udpchecksum (physaddress, USERDATA_LENGTH, template) ;
}
```

Ex. 1005.007 (Erickson) at 7:21-33;
Paper 45 (241 Reply) at 5-6;
Ex. 1003.115 (Horst Decl.) at A-5;
Paper 4 (241 Petition) at 53-54.

The prior art combination teaches header validation on the adapter



The header and data should be separately checksummed, for two reasons. First, to make it possible to checksum the header but not the data. Second, to verify that the header is correct before starting to copy the data into user space. It is desirable to do the data checksum at the time the data are copied to user space, but if the header is incorrect, the copy may be to the wrong process. To avoid an incorrect copy but to allow the data checksum to be done during copying, it is essential that the two checksums be separate.

Ex. 1006.589 (Tanenbaum96);
Ex. 1003.059-.060 (Horst Decl.) at ¶ 100;
Paper 45 (241 Reply) at 6.

There is “no reason to interrupt the processing of the host computer”

Trials@uspto.gov
571-272-7822

Paper 11
Entered: November 30, 2017

Regarding the “without interrupt” requirement of claim 17, all processing to generate headers for packets to be sent from the network interface device of Erickson is performed by the processing capability of Erickson’s network interface device with no reason to interrupt the processing of the host computer requesting the transmission.

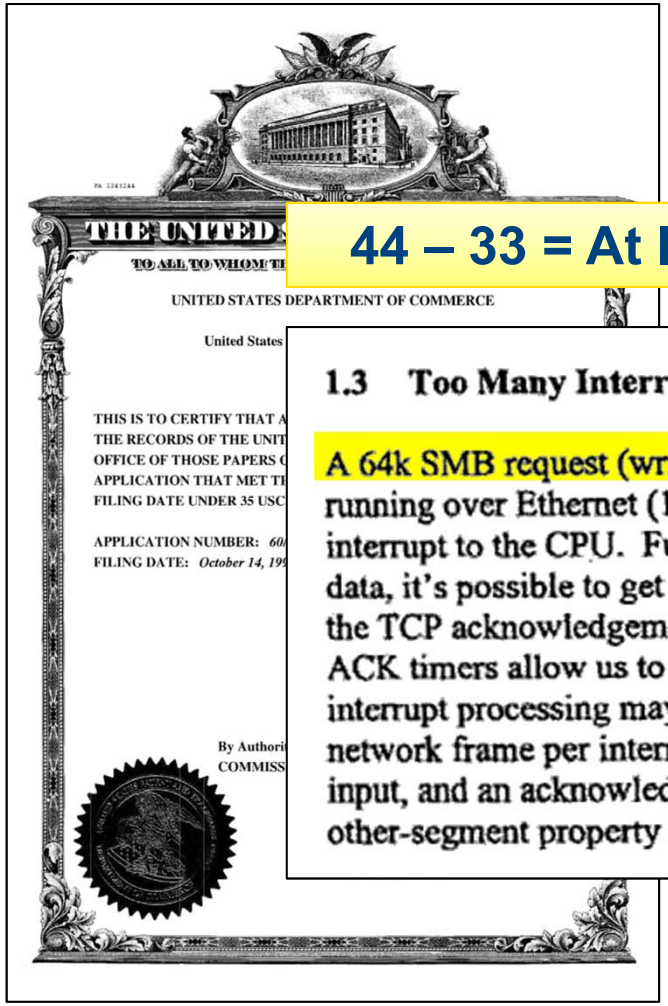
DECISION
Institution of *Inter Partes* Review
37 C.F.R. § 42.108

I. INTRODUCTION

Intel Corporation (“Petitioner”) requests *inter partes* review of claims all claims (1–24) of U.S. Patent No. 7,337,241 B2 (“the ’241 patent,” Ex.

Paper 11 (Institution Decision) at 19;
Paper 41 (241 Reply) at 17;
Ex. 1003.095-.096 (Horst Decl.) at ¶ 159.

Priority application admits using fewer than 1 interrupt per packet was known



44 – 33 = At least 11 segments with no interrupts

1.3 Too Many Interrupts

A 64k SMB request (write or read-reply) is typically made up of 44 TCP segments when running over Ethernet (1500 byte MTU). Each of these segments may result in an interrupt to the CPU. Furthermore, since TCP must acknowledge all of this incoming data, it's possible to get another 44 transmit-complete interrupts as a result of sending out the TCP acknowledgements. While this is possible, it is not terribly likely. Delayed ACK timers allow us to acknowledge more than one segment at a time. And delays in interrupt processing may mean that we are able to process more than one incoming network frame per interrupt. Nevertheless, even if we assume 4 incoming frames per input, and an acknowledgement for every 2 segments (as is typical per the ACK-every-other-segment property of TCP), we are still left with 33 interrupts per 64k SMB request.

Ex. 1031.006 (1997 Provisional); Paper 45 (241 Reply) at 6-7.

Alteon teaches using fewer than one interrupt per packet

Gigabit Ethernet Technical Brief

Using an intelligent adapter with an onboard RISC-based processor specially designed for embedded application processing, Alteon's Gigabit Ethernet technology not only reduces the number of times data is copied among processing entities, it allows a single interrupt to be issued for multiple data packets—radically altering the ratio of interrupts to packets, and eliminating the scalability problems inherent in older adapter designs.

Alteon Gigabit Ethernet adapters have an interrupt timer that determines when to interrupt a host CPU. This allows a single interrupt to be issued for multiple data packets that are sent into the operating system buffer space. It also allows for "adaptive" interrupts; that is, the NIC can alter the number of interrupts issued per second based on network usage.

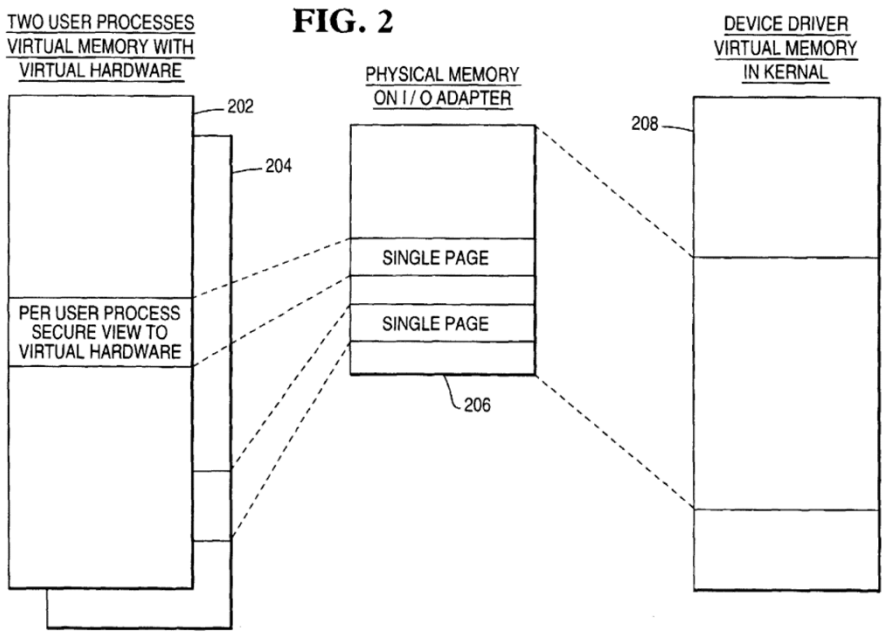
Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119

1-408-574-5500

First Edition
September 1996

Ex. 1033.022-.023 (Alteon);
Paper 45 (241 Reply) at 5.

Erickson teaches transfer without interrupts using polling and snooping



"Incoming data is then written to the virtual memory and detected by polling or "snooping" hardware."

Ex. 1005.003, -.012 (Erickson) at Fig. 2, 8:50-52;
Ex. 1223.025-.026 (Horst Reply Decl.) at ¶¶ 50-52;
Paper 45 (241 Reply) at 6.

Dr. Horst explains that snooping and polling do not involve interrupts

UNITED STATES PATENT AND TRADEMARK OFFICE

hardware'." Ex. 1005 at 8:50-52. Erickson clearly describes how the snooping works to allow the adapter and host to read and write portions of each other's memory, and this type of snooping does not involve interrupts. Erickson's

Title: FAST

DECLAR
REPLY T

software, then another context switch back to user space. Instead, Erickson's polling loop allows the user code to directly poll the STATUS register without any involvement from the operating system.

Mail Stop
Patent Trial
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

[1] Cavium, Inc., which filed a Petition in Case IPR2017-01728, has been joined as a petitioner in this proceeding. Wistron Corporation, who filed a Petition in Case IPR2018-00328, has been joined as a petitioner in this proceeding.

Ex. 1223.025 (Horst Reply Decl.) at ¶ 50;
Ex. 1005.012 (Erickson) at 6:25-31;
Paper 45 (241 Reply) at 6.

241 Patent: Disputes (Receive Claims)

a) The combination of Erickson, Tanenbaum96 and Alteon discloses the limitations of claims **1**, 2-8, 18, 22, and 23 of the 241 Patent

i. The prior art discloses validation of network and transport layer headers "without an interrupt dividing the processing" (claim **1**)

ii. <u>The prior art discloses sending the data from each packet to a destination in memory without sending any of the headers (claim 1)</u>

iii. The prior art discloses processing MAC layer headers without an interrupt (claim 2)

iv. The prior art discloses processing an upper layer header by a second mechanism (claim 3)

v. The prior art discloses sorting the packets by classifying each as having IP and TCP headers (claim 6)

241 Patent: Claim 1

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent:**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451
4,991,133	A	2/1991	Davis et al.	

(Continued)

FOREIGN PATENT DOCUM

WO WO/98/19412 5/1998

(Continued)

OTHER PUBLICATION

Internet pages entitled "Hardware Assisted P
(which Eugene Feinberg is working on). 1 pa
1998.

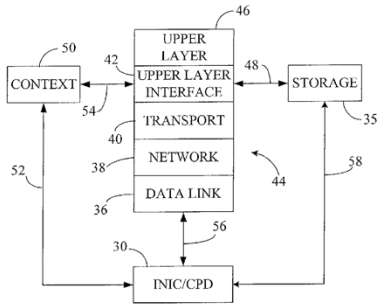
(Continued)

Primary Examiner—David Wiley
(74) *Attorney, Agent, or Firm*—Mark La
Law Group LLP

(57) **ABSTRACT**

A system for protocol processing in a com
an intelligent network interface card (INI
tion processing device (CPD) associated
puter. The INIC provides a fast-path th
processing for most large multi-packet
accelerating data communication. The IN
host for those message packets that are c
ing by host software layers. A communica
for a message is defined that allows DMA
INIC to move data, free of headers, dir
destination or source in the host. The cont
INIC as a communication control block (
passed back to the host for message proc
The INIC contains specialized hardware
much faster at their specific tasks than
CPU. A preferred embodiment includes
processors with separate processors de
receive and management processing, wit
munication for four fast Ethernet nodes.

24 Claims, 89 Drawing Sh



1. A method for network communication, the method comprising:

receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;

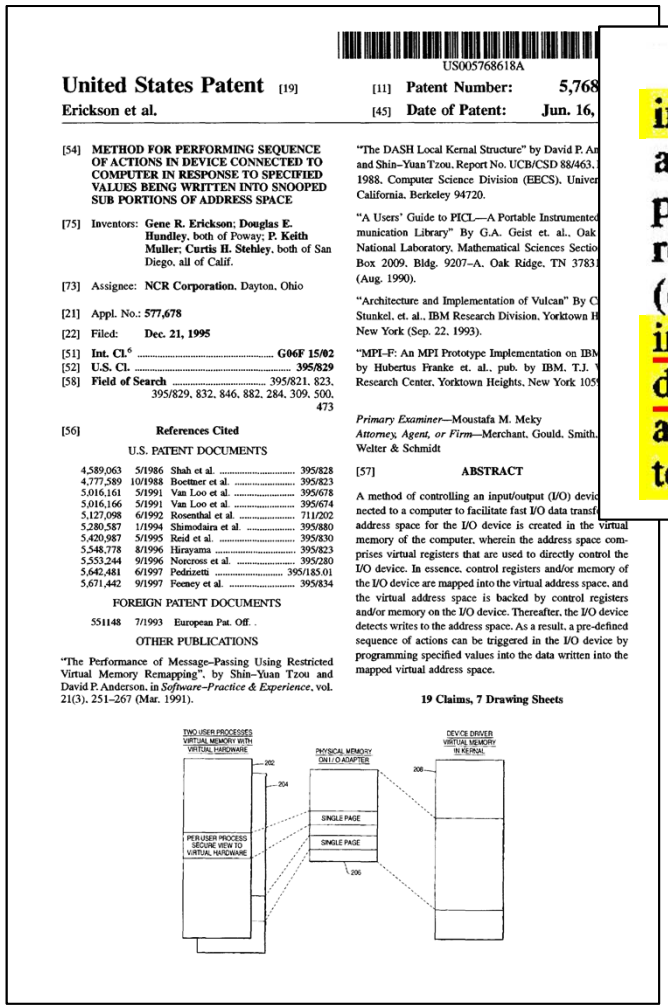
processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;

sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;

sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application without sending any of the media access control layer headers, network layer

Ex. 1001.142 (241 Patent), Claim 1.

Erickson transfers data to applications in the host directly without headers



US005768618A

United States Patent [19]

Erickson et al.

[11] Patent Number: 5,768

[45] Date of Patent: Jun. 16,

[54] METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE

[75] Inventors: Gene R. Erickson; Douglas E. Hundley, both of Poway; P. Keith Muller; Curtis H. Stehley, both of San Diego, all of Calif.

[73] Assignee: NCR Corporation, Dayton, Ohio

[21] Appl. No.: 577,678

[22] Filed: Dec. 21, 1995

[51] Int. Cl.[6] G06F 15/02

[52] U.S. Cl. 395/829

[58] Field of Search 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] References Cited

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message–Passing Using Restricted Virtual Memory Remapping", by Shin–Yuan Tzou and David P. Anderson, in *Software–Practice & Experience*, vol. 21(3), 251–267 (Mar. 1991).

"The DASH Local Kernal Structure" by David P. An and Shin–Yuan Tzou, Report No. UCB/CSD 88/463, 1988, Computer Science Division (EECS), Univer California, Berkeley 94720.

"A Users' Guide to PICL—A Portable Instrumented munication Library" By G.A. Geist et. al., Oak National Laboratory, Mathematical Sciences Sectio Box 2009, Bldg. 9207–A, Oak Ridge, TN 3783 (Aug. 1990).

"Architecture and Implementation of Vulcan" By C Stunkel, et. al., IBM Research Division, Yorktown H New York (Sep. 22, 1993).

"MPI–F: An MPI Prototype Implementation on IBM by Hubertus Franke et. al., pub. by IBM. T.J. Research Center, Yorktown Heights, New York 105

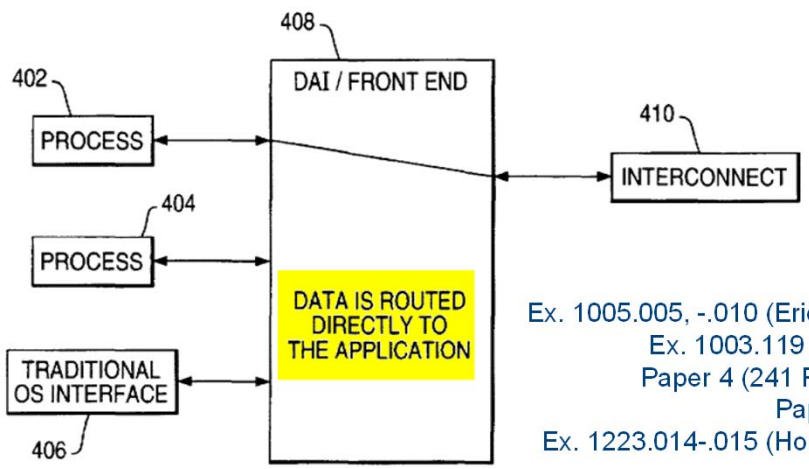
Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Welter & Schmidt

[57] ABSTRACT

A method of controlling an input/output (I/O) devic nected to a computer to facilitate fast I/O data transf address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

FIG. 4 is a block diagram describing a direct application interface (DAI) and routing of data between processes and an external data connection which is compatible with the present invention. Processes **402** and **404** transmit and receive information directly to and from an interconnect **410** (e.g., I/O device adapter) through the DAI interface **408**. The information coming from the interconnect **410** is routed directly to a process **402** or **404** by use of virtual hardware and registers, rather than using a traditional operating system interface **406**.



Ex. 1005.005, -.010 (Erickson) at Fig. 4, 5:6-14;
Ex. 1003.119 (Horst Decl.) at A-9-10;
Paper 4 (241 Petition) at 56-57 56-58;
Paper 45 (241 Reply) at 7;
Ex. 1223.014-.015 (Horst Reply Decl.) at ¶ 29.

Alteon transfers data to applications in the host without headers

Gigabi
Techni

Achieving

ALTEON NETWORKS

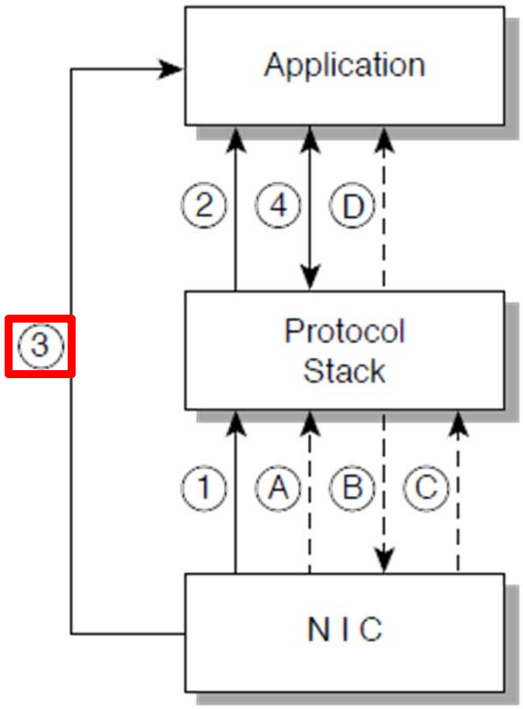
Alteon Networks, Inc.
6351 San Ignacio Avenue
San Jose, CA 95119

1-408-574-5500

First Edition
September 1996

Table 4. Steps in Data Reception using Second Generation NIC

Data Steps	Control Steps
1. The NIC moves the first 64 bytes of the packet to the protocol stack through a pre-allocated buffer. The first 64 bytes includes the header information and some data.	A. The NIC notifies the stack that it has moved 64 bytes of data by issuing an interrupt.
	B. The protocol stack analyzes the headers and tells the NIC where in application memory to put the remaining data from the packet.
2. The protocol stack moves any data from the initial 64 bytes (minus the header) to the application memory.	
3. The NIC moves the rest of the data into application memory. If application memory is not accessible, then an intermediate buffer is used and the data is copied to the application memory by the host in Step 4.	C. The NIC tells the stack that it has finished moving the rest of the data packet into application memory by issuing an interrupt.
4. The protocol stack performs a checksum on the packet in the application memory space.	D. The protocol stack informs the application that data has arrived.



Paper 4 (241 Petition) at 57;
Ex. 1003.124 (Horst Decl.) at A-15;
Ex. 1033.021 (Alteon).

241 Patent: Disputes (Receive Claims)

a. The combination of Erickson, Tanenbaum96 and Alteon discloses the limitations of claims **1**, 2-8, 18, 22, and 23 of the 241 Patent

i. The prior art discloses validation of network and transport layer headers "without an interrupt dividing the processing" (claim **1**)

ii. The prior art discloses sending the data from each packet to a destination in memory without sending any of the headers (claim **1**)

iii. <u>The prior art discloses processing MAC layer headers without an interrupt (claim 2)</u>

iv. The prior art discloses processing an upper layer header by a second mechanism (claim 3)

v. The prior art discloses sorting the packets by classifying each as having IP and TCP headers (claim 6)

241 Patent: Claim 2

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

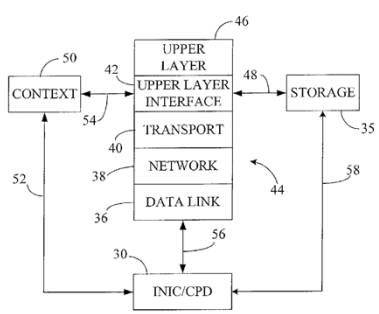
(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451

for a message is defined that allows DMA controllers of the INIC to move data, free of headers, directly to or from a destination or source in the host. The context is stored in the INIC as a communication control block (CCB) that can be passed back to the host for message processing by the host. The INIC contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors with separate processors devoted to transmit, receive and management processing, with full duplex communication for four fast Ethernet nodes.

24 Claims, 89 Drawing Sheets

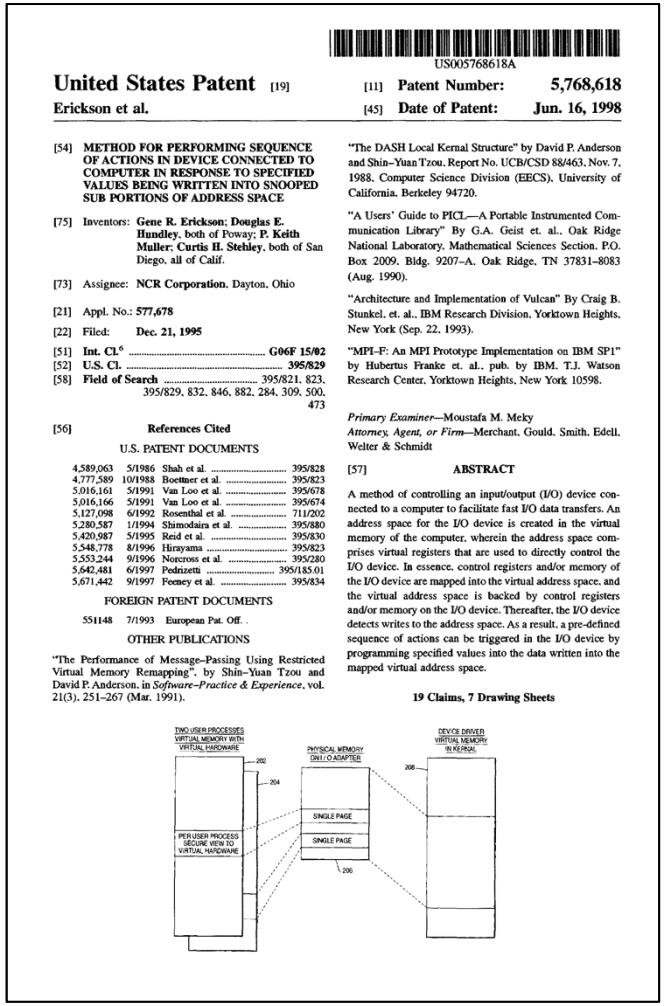


2. The method of claim **1**, wherein processing the packets by a first mechanism further comprises:

processing the media access control layer header for each packet without an interrupt dividing the processing of the media access control layer header and the network layer header.

Ex. 1001.142 (241 Patent), Claim 2.

Erickson teaches that the MAC layer header is processed on the adapter



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson; Douglas E. Hundley**, both of Poway; **P. Keith Muller; Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodairs et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

"The DASH Local Kernal Structure" by David P. Anderson and Shin-Yuan Tzou, Report No. UCB/CSD 88/463, Nov. 7, 1988, Computer Science Division (EECS), University of California, Berkeley 94720.

"A Users' Guide to PICL—A Portable Instrumented Communication Library" By G.A. Geist et. al., Oak Ridge National Laboratory, Mathematical Sciences Section, P.O. Box 2009, Bldg. 9207-A, Oak Ridge, TN 37831-8083 (Aug. 1990).

"Architecture and Implementation of Vulcan" By Craig B. Stunkel, et. al., IBM Research Division, Yorktown Heights, New York (Sep. 22, 1993).

"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Edell, Welter & Schmidt

[57] **ABSTRACT**

A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

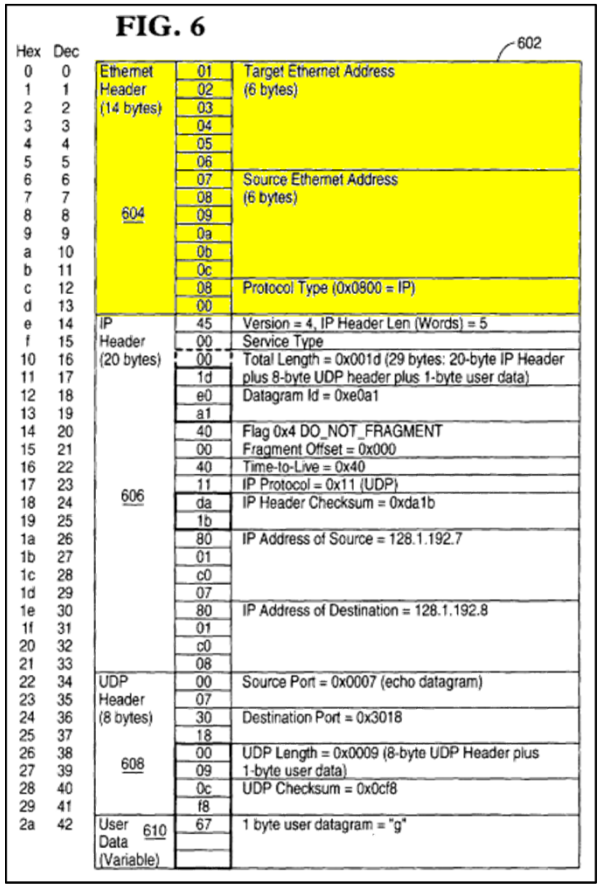


FIG. 6

602

Hex	Dec			
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Address (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 = IP)
d	13		00	
e	14	IP Header (20 bytes) 606	45	Version = 4, IP Header Len (Words) = 5
f	15		00	Service Type
10	16		00	Total Length = 0x001d (29 bytes: 20-byte IP Header plus 8-byte UDP header plus 1-byte user data)
11	17		1d	
12	18		e0	Datagram Id = 0xe0a1
13	19		a1	
14	20		40	Flag 0x4 DO_NOT_FRAGMENT
15	21		00	Fragment Offset = 0x000
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP)
18	24		da	IP Header Checksum = 0xda1b
19	25		1b	
1a	26		80	IP Address of Source = 128.1.192.7
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destination = 128.1.192.8
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 608	00	Source Port = 0x0007 (echo datagram)
23	35		07	
24	36		30	Destination Port = 0x3018
25	37		18	
26	38		00	UDP Length = 0x0009 (8-byte UDP Header plus 1-byte user data)
27	39		09	
28	40		0c	UDP Checksum = 0x0cf8
29	41		f8	
2a	42	User Data (Variable) 610	67	1 byte user datagram = "g"

Ex. 1005.007 (Erickson), Fig. 6.

241 Patent: Disputes (Receive Claims)

a) The combination of Erickson, Tanenbaum96 and Alteon discloses the limitations of claims **1**, 2-8, 18, 22, and 23 of the 241 Patent

i. The prior art discloses validation of network and transport layer headers "without an interrupt dividing the processing" (claim **1**)

ii. The prior art discloses sending the data from each packet to a destination in memory without sending any of the headers (claim **1**)

iii. The prior art discloses processing MAC layer headers without an interrupt (claim 2)

iv. <u>The prior art discloses processing an upper layer header by a second mechanism (claim 3)</u>

v. The prior art discloses sorting the packets by classifying each as having IP and TCP headers (claim 6)

241 Patent: Claim 3

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

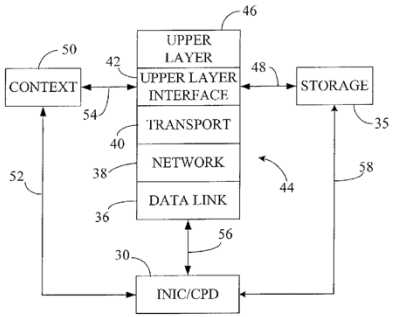
(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451

puter. The INIC provides a fast-path that avoids protocol processing for most large multi-packet messages, greatly accelerating data communication. The INIC also assists the host for those message packets that are chosen for processing by host software layers. A communication control block for a message is defined that allows DMA controllers of the INIC to move data, free of headers, directly to or from a destination or source in the host. The context is stored in the INIC as a communication control block (CCB) that can be passed back to the host for message processing by the host. The INIC contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors with separate processors devoted to transmit, receive and management processing, with full duplex communication for four fast Ethernet nodes.

24 Claims, 89 Drawing Sheets

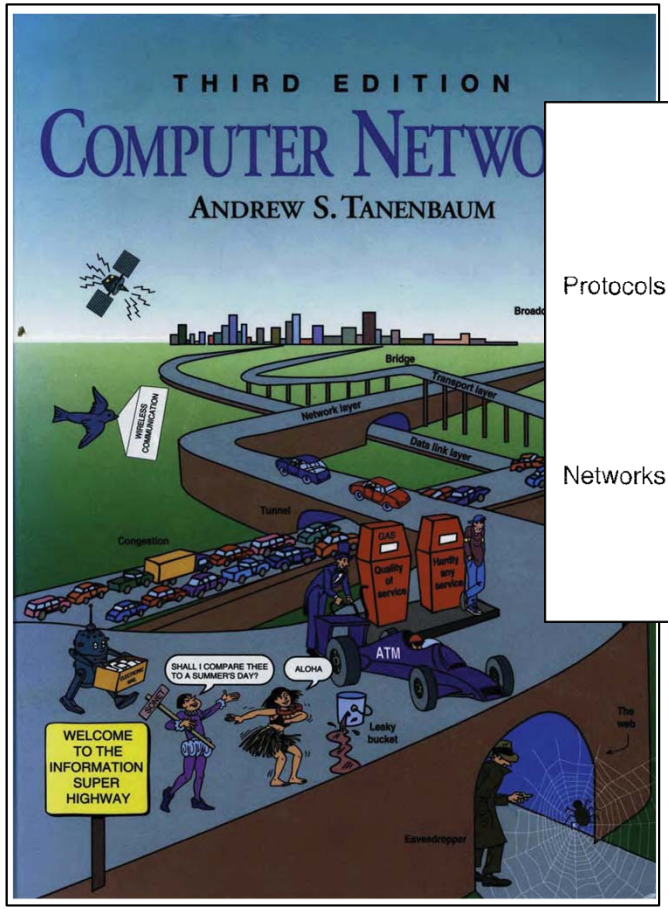


3. The method of claim **1**, further comprising:

processing an upper layer header of at least one of the packets by a second mechanism, thereby determining the destination, wherein the upper layer header corresponds to a protocol layer above the transport layer.

Ex. 1001.142 (241 Patent), Claim 3.

Tanenbaum96: The processing of application headers



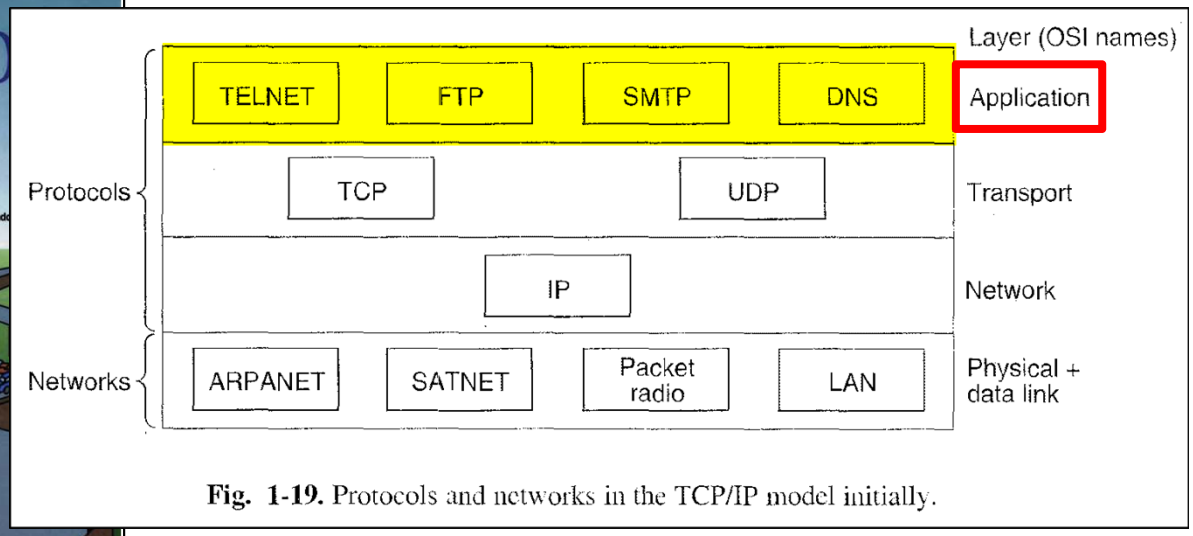


Fig. 1-19. Protocols and networks in the TCP/IP model initially.

Paper 45 (241 Reply) at 8;
Paper 4 (241 Petition) at 60;
Ex. 1006.055 (Tanenbaum96).

241 Patent: Disputes (Receive Claims)

a) The combination of Erickson, Tanenbaum96 and Alteon discloses the limitations of claims **1**, 2-8, 18, 22, and 23 of the 241 Patent

i. The prior art discloses validation of network and transport layer headers "without an interrupt dividing the processing" (claim **1**)

ii. The prior art discloses sending the data from each packet to a destination in memory without sending any of the headers (claim **1**)

iii. The prior art discloses processing MAC layer headers without an interrupt (claim 2)

iv. The prior art discloses processing an upper layer header by a second mechanism (claim 3)

v. <u>The prior art discloses sorting the packets by classifying each as having IP and TCP headers (claim 6)</u>

241 Patent: Claim 6

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451
4,991,133	A	2/1991	Davis et al.	364/900

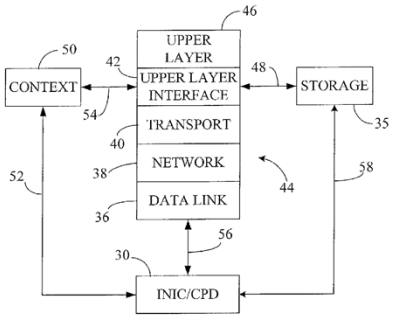
(Continued)

FOREIGN PATENT DOCUMENTS

WO	WO/98/19412	5/1998

host for those message packets that are chosen for processing by host software layers. A communication control block for a message is defined that allows DMA controllers of the INIC to move data, free of headers, directly to or from a destination or source in the host. The context is stored in the INIC as a communication control block (CCB) that can be passed back to the host for message processing by the host. The INIC contains specialized hardware circuits that are much faster at their specific tasks than a general purpose CPU. A preferred embodiment includes a trio of pipelined processors with separate processors devoted to transmit, receive and management processing, with full duplex communication for four fast Ethernet nodes.

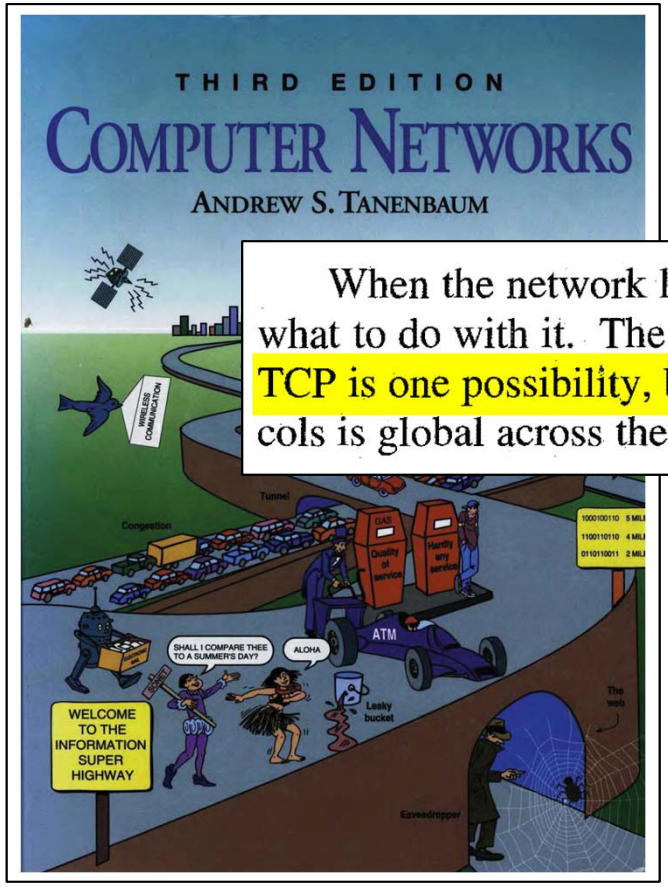
24 Claims, 89 Drawing Sheets



6. The method of claim **1**, wherein sorting the packets includes classifying each of the packets of the first type as having an Internet Protocol (IP) header and a Transmission Control Protocol (TCP).

Ex. 1001.143 (241 Patent), Claim 6.

Tanenbaum96: Parsing the header to determine packet's protocol



When the network layer has assembled a complete datagram, it needs to know what to do with it. The *Protocol* field tells it which transport process to give it to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet and is defined in RFC 1700.

Paper 4 (241 Petition) at 63-64;
Ex. 1006.433 (Tanenbaum96).

241 Patent: Disputes (Transmit Claims)

b) Erickson in view of Tanenbaum96 discloses the limitations of claims **9**, 10-16, **17**, 19-21, and 21 of the 241 Patent

i. **The prior art discloses "prepending the MAC, network, and transport layer headers at one time as a sequence of bits " (claim 9)**

ii. The prior art discloses prepending each packet header without an interrupt dividing the prepending of the MAC, IP, and TCP headers (claim **17**)

iii. The prior art discloses dividing the data into multiple segments and prepending a packet header to each of the segments by a second processor/mechanism (claims **9** and **17**)

241 Patent: Claim 19

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

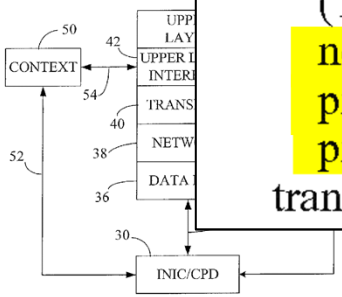
(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538	A	12/1982	Johnson et al.	364/200
4,485,455	A	11/1984	Boone et al.	364/900
4,485,460	A	11/1984	Stambaugh	365/203
4,589,063	A	5/1986	Shah et al.	710/8
4,700,185	A	10/1987	Balph et al.	370/451



9. A method for communicating information over a network, the method comprising:

obtaining data from a source in memory allocated by a first processor;

dividing the data into multiple segments;

prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and

transmitting the packets to the network.

Ex. 1001.143 (241 Patent), Claim 19.

Erickson teaches the use of a template to create headers

FIG. 6

602

Hex	Dec			
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Address (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 = IP)
d	13		00	
e	14	IP Header (20 bytes) 606	45	Version = 4, IP Header Len (Words) = 5
f	15		00	Service Type
10	16		00	Total Length = 0x001d (29 bytes: 20-byte IP Header
11	17		1d	plus 8-byte UDP header plus 1-byte u
12	18		e0	Datagram Id = 0xe0a1
13	19		a1	
14	20		40	Flag 0x4 DO_NOT_FRAGMENT
15	21		00	Fragment Offset = 0x000
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP)
18	24		da	IP Header Checksum = 0xda1b
19	25		1b	
1a	26		80	IP Address of Source = 128.1.192.7
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destination = 128.1.192
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 608	00	Source Port = 0x0007 (echo datagram
23	35		07	
24	36		30	Destination Port = 0x3018
25	37		18	
26	38		00	UDP Length = 0x0009 (8-byte UDP Header plus
27	39		09	1-byte user data)
28	40		0c	UDP Checksum = 0x0cf8
29	41		f8	
2a	42	User Data (Variable) 610	67	1 byte user datagram = "g"

It would have been obvious to prepend the populated header to the data at one time

In the present application, the access privileges given to the user processes are very narrow. Each user process has basically pre-negotiated almost everything about the datagram **602**, except the actual user data **610**. This means most of the fields in the three header areas **604**, **606**, and **608** are predetermined.

Ex. 1005.007, .011 (Erickson) at Fig. 6, 6:57-62
Paper 45 (241 Reply) at 13-14;
Paper 4 (241 Petition) at 73-74;
Ex. 1003.075-79,.136-.141 (Horst Decl.) at ¶¶ 131-136, A-16 – A-31.

Dr. Almeroth's interpretation would result in an invalid packet

3

2

1

708 plus the user data." (*Id.* at 8:24-26.) When building a packet for transmission, *Erickson* thus appears to first prepends the Ethernet header 704 to the user data, then prepends the IP header 706 to the Ethernet header, and finally prepends the UDP header 708 to the IP header. In other words, due to the serial nature of the execution of the udpscript() procedure, *Erickson* appears to also likely builds the UDP datagram ***in a traditional serial fashion***, rather than prepending the MAC, transport, and network headers "at one time" as required by claim 9.

FIG. 6

1

2

3

	Hex	Dec	602
Ethernet Header (14 bytes) 604	01		Target Ethernet Address (6 bytes)
	02		
	03		
	04		
	05		
	06		
	07		Source Ethernet Address (6 bytes)
	08		
	09		
	0a		
	0b		
	0c		
	08		Protocol Type (0x0800 = IP)
	00		
IP Header (20 bytes) 606	45		Version = 4, IP Header Len (Words) = 5
	00		Service Type
	00		Total Length = 0x001d (29 bytes: 20-byte IP Header plus 8-byte UDP header plus 1-byte user data)
	1d		
	e0		Datagram Id = 0xe0a1
	a1		
	40		Flag 0x4 DO_NOT_FRAGMENT
	00		Fragment Offset = 0x000
	40		Time-to-Live = 0x40
	11		IP Protocol = 0x11 (UDP)
	da		IP Header Checksum = 0xda1b
	1b		
	80		IP Address of Source = 128.1.192.7
	01		
	c0		
	07		
	80		IP Address of Destination = 128.1.192.8
	01		
	c0		
	08		
UDP Header (8 bytes) 608	00		Source Port = 0x0007 (echo datagram)
	07		
	30		Destination Port = 0x3018
	18		
	00		UDP Length = 0x0009 (8-byte UDP Header plus 1-byte user data)
	09		
	0c		UDP Checksum = 0x0cf8
	f8		
User Data 610 (Variable)	67		1 byte user datagram = "g"

Ex. 2026.068 (Almeroth) at ¶ 134;
Ex. 1005.007 (Erickson) Fig. 6, 7:50-64;
Paper 45 (241 Reply) at 13-14.

Erickson teaches the use of a template to create headers

Trials@uspto.gov
571-272-7822

Paper 11
Entered: November 30, 2017

UNITED STATES PATENT AND TRADEM

BEFORE THE PATENT TRIAL AND APP

INTEL CORPORATION,
Petitioner,

v.

ALACRITECH, INC.,
Patent Owner.

Case IPR2017-01392
Patent 7,337,241 B2

Before STEPHEN C. SIU, DANIEL N. FISHMAN, a
WILLIAM M. FINK, *Administrative Patent Judges*.

FISHMAN, *Administrative Patent Judge*.

DECISION
Institution of *Inter Partes* Review
37 C.F.R. § 42.108

I. INTRODUCTION

Intel Corporation ("Petitioner") requests *inter partes* review of claims
all claims (1–24) of U.S. Patent No. 7,337,241 B2 ("the '241 patent," Ex.

packet. *Id.* at 7:46–47; *see also id.* at 6:48–56. We agree with Petitioner that the header and data to be transmitted, both stored in the memory of the network interface device, would be combined in one of two obvious manners—either the header is prepended to the data or the data is appended to the header. Given the small number of known solutions to combining the header and data, it would have been obvious to try prepending the header to the data to transmit the packet. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398,

Paper 11 (Institution Decision) at 17-18.

241 Patent: Disputes (Transmit Claims)

b) Erickson in view of Tanenbaum96 discloses the limitations of claims **9**, 10-16, **17**, 19-21, and 21 of the 241 Patent

i. The prior art discloses "prepending the MAC, network, and transport layer headers at one time as a sequence of bits " (claim **9**)

ii. <u>The prior art discloses prepending each packet header without an interrupt dividing the prepending of the MAC, IP, and TCP headers (claim 17)</u>

iii. The prior art discloses dividing the data into multiple segments and prepending a packet header to each of the segments by a second processor/mechanism (claims **9** and **17**)

241 Patent: Claim 17

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

(75) Inventors: **Laurence B. Boucher**, Saratoga, CA (US); **Stephen E. J. Blightman**, San Jose, CA (US); **Peter K. Craft**, San Francisco, CA (US); **David A. Higgen**, Saratoga, CA (US); **Clive M. Philbrick**, San Jose, CA (US); **Daryl D. Starr**, Milpitas, CA (US)

(73) Assignee: **Alacritech, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 879 days.

(21) Appl. No.: **10/260,878**

(22) Filed: **Sep. 27, 2002**

(65) **Prior Publication Data**

US 2004/0064578 A1 Apr. 1, 2004

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/250**

(58) **Field of Classification Search** 709/250
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538 A	12/1982	Johnson et al.	364/200
4,485,455 A	11/1984	Boone et al.	364/900
4,485,460 A	11/1984	Stambaugh	365/203
4,589,063 A	5/1986	Shah et al.	710/8
4,700,185 A	10/1987	Balph et al.	370/451
4,991,133 A	2/1991	Dav	

(Continu

FOREIGN PATENT

WO WO/98/19412 5

(Continu

OTHER PUBLI

Internet pages entitled "Hardware A
(which Eugene Feinberg is working
1998.

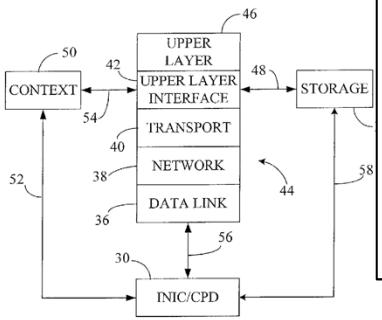
(Continu

Primary Examiner—David Wile
(74) *Attorney, Agent, or Firm*—
Law Group LLP

(57) **ABSTRA**

A system for protocol processing
an intelligent network interface c
tion processing device (CPD) as
puter. The INIC provides a fast
processing for most large multi
accelerating data communication
host for those message packets t
ing by host software layers. A co
for a message is defined that allo
INIC to move data, free of head
destination or source in the host.
INIC as a communication contro
passed back to the host for messa
The INIC contains specialized
much faster at their specific tas
CPU. A preferred embodiment i
processors with separate proces
receive and management process
munication for four fast Etherne

24 Claims, 89 Dra



17. A method for communicating information over a network, the method comprising:

providing, by a first mechanism, a block of data and a Transmission Control Protocol (TCP) connection;

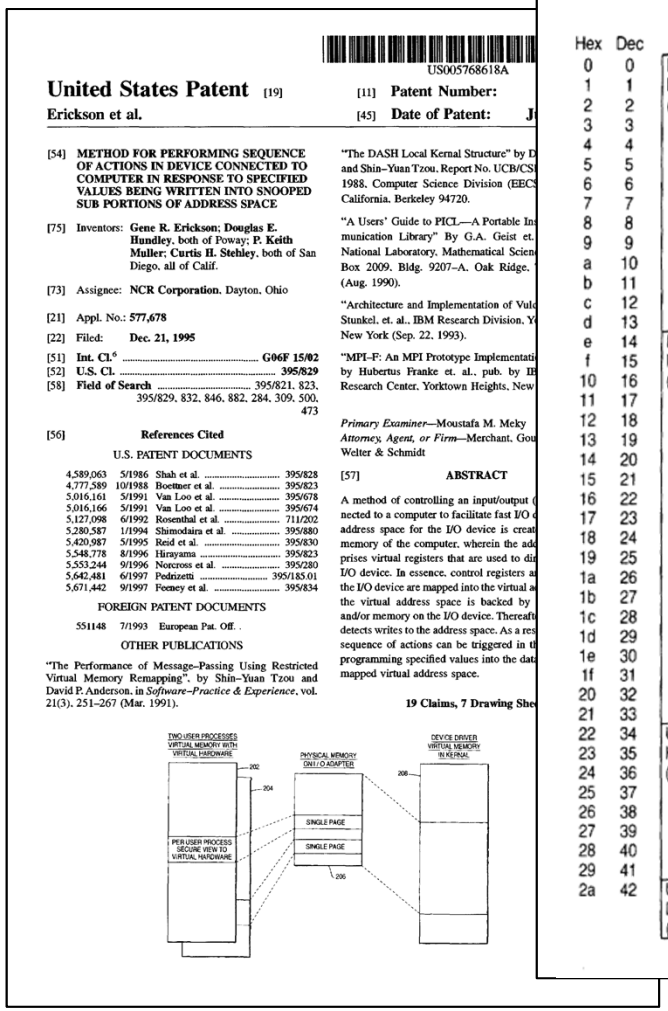
dividing, by a second mechanism, the block of data into multiple segments;

prepending, by the second mechanism, an outbound packet header to each of the segments, thereby forming an outbound packet corresponding to each segment, the outbound packet header containing an outbound media access control layer header, an outbound Internet Protocol (IP) header and an outbound TCP header, wherein the prepending of each outbound packet header occurs without an interrupt dividing the prepending of the outbound media access control layer header, the outbound (IP) header and the outbound TCP header; and

transmitting the outbound packets to the network.

Ex. 1001.143 (241 Patent), Claim 17.

The MAC header is part of the prepopulated header template



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number:**

[45] **Date of Patent:** J

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson; Douglas E. Hundley**, both of Poway; **P. Keith Muller; Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

"The DASH Local Kernal Structure" by D and Shin-Yuan Tzou, Report No. UCB/CS 1988, Computer Science Division (EECS California, Berkeley 94720.

"A Users' Guide to PICL—A Portable In munication Library" By G.A. Geist et. National Laboratory, Mathematical Scien Box 2009, Bldg. 9207-A, Oak Ridge, (Aug. 1990).

"Architecture and Implementation of Vul Stunkel, et. al., IBM Research Division, Y New York (Sep. 22, 1993).

"MPI-F: An MPI Prototype Implementati by Hubertus Franke et. al., pub. by IB Research Center, Yorktown Heights, New

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gou Welter & Schmidt

[57] **ABSTRACT**

A method of controlling an input/output (nected to a computer to facilitate fast I/O address space for the I/O device is creat memory of the computer, wherein the add prises virtual registers that are used to dir I/O device. In essence, control registers a the I/O device are mapped into the virtual a the virtual address space is backed by and/or memory on the I/O device. Thereaft detects writes to the address space. As a res sequence of actions can be triggered in t programming specified values into the dat mapped virtual address space.

19 Claims, 7 Drawing She

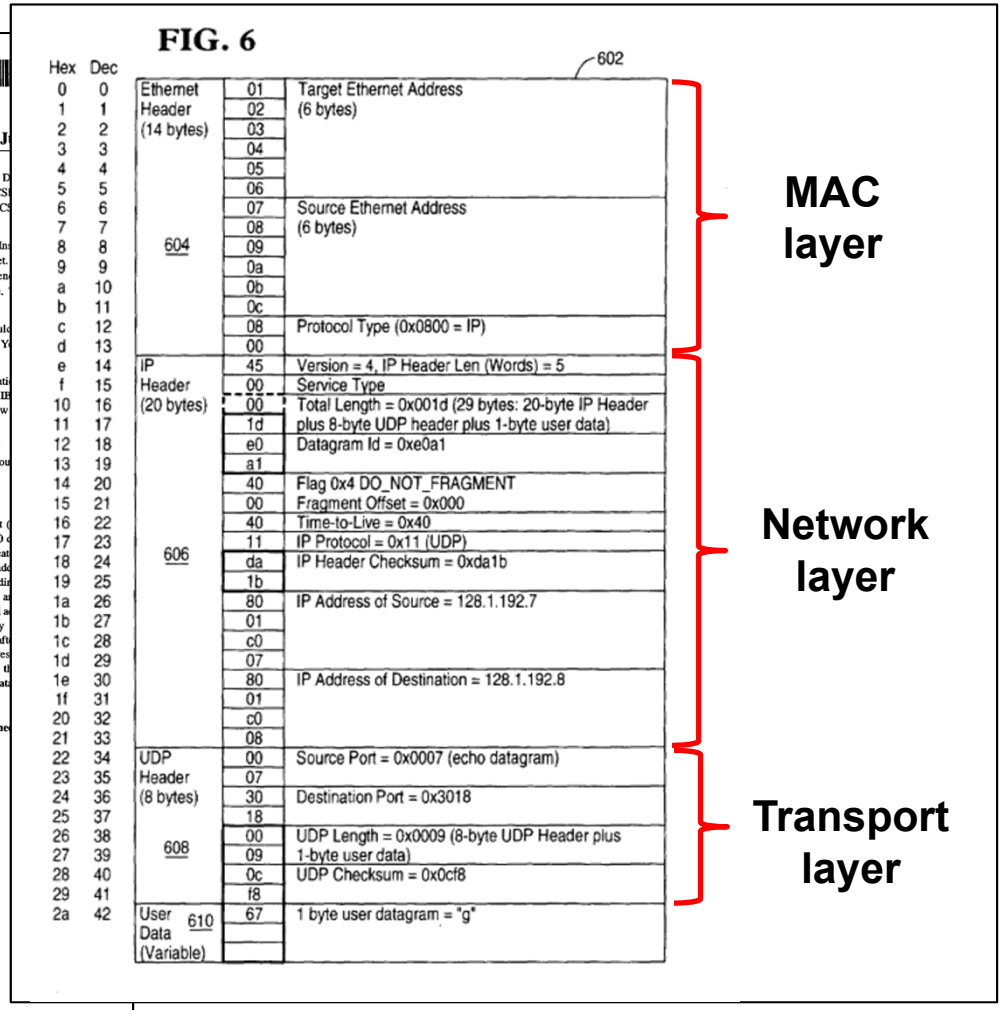


FIG. 6

602

Hex	Dec			
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Address (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 = IP)
d	13		00	
e	14	IP Header (20 bytes) 606	45	Version = 4, IP Header Len (Words) = 5
f	15		00	Service Type
10	16		00	Total Length = 0x001d (29 bytes: 20-byte IP Header plus 8-byte UDP header plus 1-byte user data)
11	17		1d	
12	18		e0	Datagram Id = 0xe0a1
13	19		a1	
14	20		40	Flag 0x4 DO_NOT_FRAGMENT
15	21		00	Fragment Offset = 0x000
16	22		40	Time-to-Live = 0x40
17	23		11	IP Protocol = 0x11 (UDP)
18	24		da	IP Header Checksum = 0xda1b
19	25		1b	
1a	26		80	IP Address of Source = 128.1.192.7
1b	27		01	
1c	28		c0	
1d	29		07	
1e	30		80	IP Address of Destination = 128.1.192.8
1f	31		01	
20	32		c0	
21	33		08	
22	34	UDP Header (8 bytes) 608	00	Source Port = 0x0007 (echo datagram)
23	35		07	
24	36		30	Destination Port = 0x3018
25	37		18	
26	38		00	UDP Length = 0x0009 (8-byte UDP Header plus 1-byte user data)
27	39		09	
28	40		0c	UDP Checksum = 0x0cf8
29	41		f8	
2a	42	User Data (Variable) 610	67	1 byte user datagram = "g"

Ex. 1005.007 (Erickson) Fig. 6;
Paper 4 (241Petition) at 50-51.

There is “no reason to interrupt the processing of the host computer”

> Regarding the “without interrupt” requirement of claim 17, all processing to generate headers for packets to be sent from the network interface device of Erickson is performed by the processing capability of Erickson’s network interface device with no reason to interrupt the processing of the host computer requesting the transmission.

Paper 11 (Institution Decision) at 19;
Paper 45 (241 Reply) at 14;
Ex. 1223.016 (Horst Reply Decl.) at ¶ 31.

241 Patent: Disputes (Transmit Claims)

b) Erickson in view of Tanenbaum96 discloses the limitations of claims **9**, 10-16, **17**, 19-21, and 21 of the 241 Patent

i. The prior art discloses "prepending the MAC, network, and transport layer headers at one time as a sequence of bits " (claim **9**)

ii. The prior art discloses prepending each packet header without an interrupt dividing the prepending of the MAC, IP, and TCP headers (claim **17**)

iii. <u>The prior art discloses dividing the data into multiple segments and prepending a packet header to each of the segments by a second processor/mechanism (claims 9 and 17)</u>

241 Patent: Claims 9, 17

(12) **United States Patent**
Boucher et al.

(10) **Patent No.: US 7,337,241 B2**
(45) **Date of Patent: Feb. 26, 2008**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING TO A TCP CONNECTION**

4,991,133 A 2/1991 Davis et al. 364/900
(Continued)

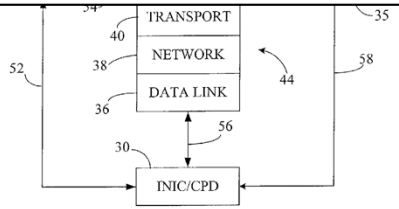
9. A method for communicating information over a network, the method comprising:

obtaining data from a source in memory allocated by a first processor;

dividing the data into multiple segments;

prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and

transmitting the packets to the network.



17. A method for communicating information over a network, the method comprising:

providing, by a first mechanism, a block of data and a Transmission Control Protocol (TCP) connection;

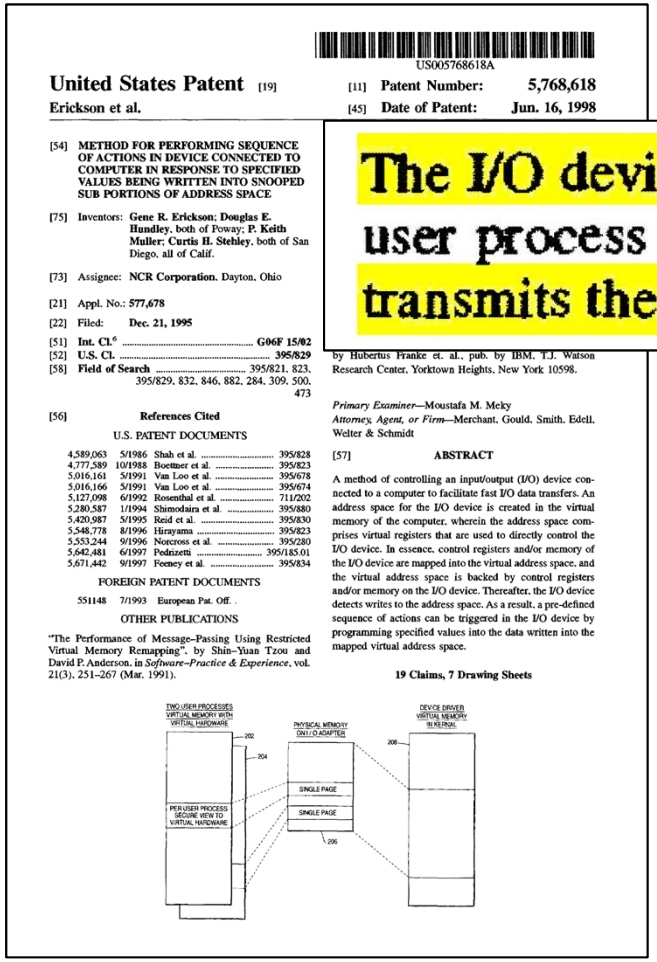
dividing, by a second mechanism, the block of data into multiple segments;

prepending, by the second mechanism, an outbound packet header to each of the segments, thereby forming an outbound packet corresponding to each segment, the outbound packet header containing an outbound media access control layer header, an outbound Internet Protocol (IP) header and an outbound TCP header, wherein the prepending of each outbound packet header occurs without an interrupt dividing the prepending of the outbound media access control layer header, the outbound (IP) header and the outbound TCP header; and

transmitting the outbound packets to the network.

Ex. 1001.143 (241 Patent), Claims 9, 17.

Erickson teaches that its interface device stores and transmits user data



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson**; **Douglas E. Hundley**, both of Poway; **P. Keith Muller**; **Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Noecross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Gould, Smith, Edell, Welter & Schmidt

[57] **ABSTRACT**

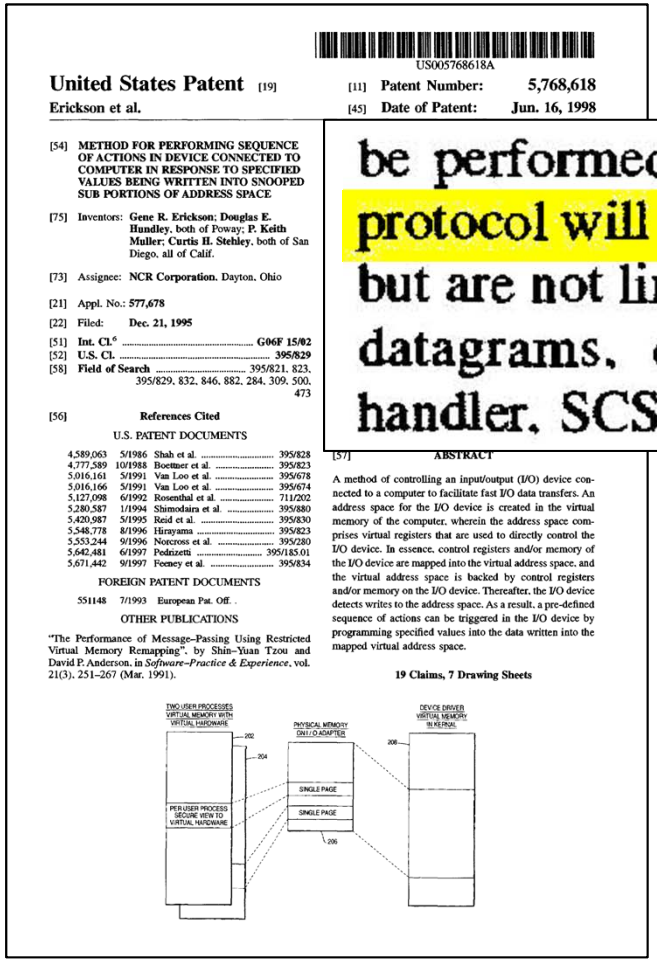
A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

The I/O device adapter stores the user data provided by the user process in the I/O device adapter's memory, and then transmits the completed UDP datagram 702 over the media.

Paper 4 (241 Petition) at 67;
Paper 45 (241 Reply) at 15-16;
Ex. 1005.012 (Erickson) at 7:39-41.

Erickson: Scripts executed by the adapter implement TCP/IP



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number: 5,768,618**

[45] **Date of Patent: Jun. 16, 1998**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson**; **Douglas E. Hundley**, both of Poway; **P. Keith Muller**; **Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Noecross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using Restricted Virtual Memory Remapping", by Shin-Yuan Tzou and David P. Anderson, in *Software-Practice & Experience*, vol. 21(3), 251-267 (Mar. 1991).

[57] **ABSTRACT**

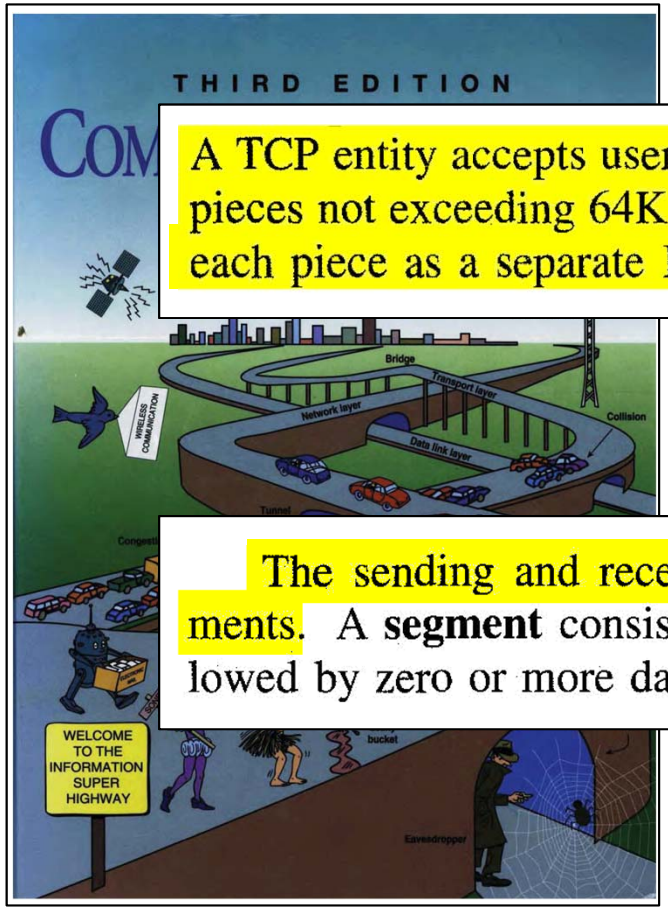
A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. In essence, control registers and/or memory of the I/O device are mapped into the virtual address space, and the virtual address space is backed by control registers and/or memory on the I/O device. Thereafter, the I/O device detects writes to the address space. As a result, a pre-defined sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

be performed based upon the protocol type. Each type of protocol will have its own script. Types of protocols include, but are not limited to, TCP/IP, UDP/IP, BYNET lightweight datagrams, deliberate shared memory, active message handler, SCSI, and File Channel

Paper 4 (241 Petition) at 73;
Paper 45 (241 Reply) at 15;
Ex. 1005.011 (Erickson) at 5:47-51.

Tanenbaum96: TCP segments data



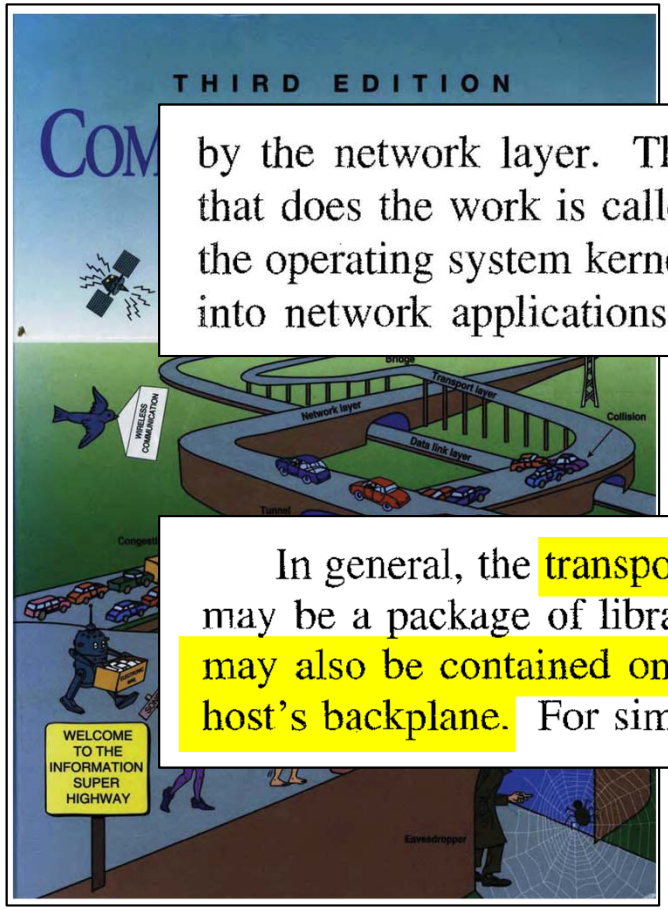
A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64K bytes (in practice, usually about 1500 bytes), and sends each piece as a separate IP datagram. When IP datagrams containing TCP data

Paper 4 (241 Petition) at 73;
Paper 45 (241 Reply) at 15-16;
Ex. 1003.149 (Horst Decl.) at A-39;
Ex. 1006.540 (Tanenbaum96).

The sending and receiving TCP entities exchange data in the form of segments. A **segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments

Paper 4 (241 Petition) at 73;
Paper 45 (241 Reply) at 15-16;
Ex. 1003.149 (Horst Decl.) at A-39;
Ex. 1006.543 (Tanenbaum96).

Tanenbaum96: Transport entity may reside on network interface



by the network layer. The hardware and/or software within the transport layer that does the work is called the **transport entity**. The transport entity can be in the operating system kernel, in a separate user process, in a library package bound into network applications, or on the network interface card. In some cases, the

Paper 4 (241 Petition) at 39;
Paper 45 (241 Reply) at 10;
Ex. 1003.150 (Horst Decl.) at A-40;
Ex. 1006.498 (Tanenbaum96).

In general, the transport entity may be part of the host's operating system or it may be a package of library routines running within the user's address space. It may also be contained on a coprocessor chip or network board plugged into the host's backplane. For simplicity, our example has been programmed as though it

Paper 45 (241 Reply) at 10;
Ex. 1003.152 (Horst Decl.) at A-42;
Ex. 1006.530 (Tanenbaum96).

241 Patent: Disputes

3. Alteon is Prior Art

a) **Alteon was available on Alteon.com before the priority date**

b) Alteon and Alteon.com were known to POSAs

c) Patent Owner Submitted a Substantively Identical version of Alteon as Prior Art

Alteon was easily accessible from Alteon.com

JUN 22 1997

12/27/2017 Alteon Tech Brief intro

http://www.alteon.com:80/techbr01.html Go

26 captures

22 Jun 1997 - 23 Oct 2017

Back Home | About Alteon | What's New | Press Room | Employment Opportunities | Products & Technology | Gigabit Resources

TECHNICAL Brief

Table of Contents

click here to DOWNLOAD the Technical Brief in PDF format

groupware, medical imaging, CAD/CAM applications, 3-D modeling, animation, video, pre-press applications, server farms, seismic processing...

The list goes on and on. The demand for high-speed network connections is proliferating at a pace almost as rapid as the speed requirements of the applications themselves. Evidence is everywhere: the rapid acceptance of 10/100 Mbps connections on today's desktop computers, Ethernet switching at the department level, and the deployment of Fast Ethernet switches in corporate backbones are a few examples of the need for faster and faster networks.

And still, bottlenecks remain. Server network connections have been limited to 100 Mbps since FDDI was shipping in volume in the late 1980's. Fast Ethernet made it easier to build internetworking products, but did not provide a faster server interface. Today, centralized servers are often configured with multiple 100 Mbps network connections to meet bandwidth requirements.
Enter Gigabit Ethernet.

Gigabit Ethernet is a new technology that will provide seamless interoperability with Ethernet and Fast Ethernet. Gigabit Ethernet transfers data at a blazingly fast

https://web.archive.org/web/19970622102901/http://www.alteon.com:80/techbr01.html 1/2

INTEL EXHIBIT 1203.001

Ex. 1203.001 (Alteon Website);
Paper 45 (241 Reply) at 3;
Ex. 1223.015 (Horst Reply Decl.) at ¶ 26.

241 Patent: Disputes

3. Alteon is Prior Art

a) Alteon was available on Alteon.com before the priority date

b) **Alteon and Alteon.com were known to POSAs**

c) Patent Owner Submitted a Substantively Identical version of Alteon as Prior Art

Dr. Horst: Alteon was well known to POSAs

NETWORKING

IBM, Alteon strike Gigabit Ethernet deal

■ Partnership will bring no-hop routing capability to non-ATM networks

By Stephen Lawson

A PARTNERSHIP between IBM and Alteon Networks, announced last week at NetWorld+Interop, in Las Vegas, will allow Alteon's pre-standard Gigabit Ethernet server adapters to route traffic directly across a network with assistance from IBM's MSS ATM route server.

Multiprotocol Switched Services (MSS) client software, available today for Asynchronous Transfer Mode (ATM) adapters, lets the adapters request route information from an MSS server and then cache that information to enable no-hop routing from source to destination. Currently, for end-stations on non-ATM network segments, the route information is cached at the nearest ATM device, which allows one-hop routing.

The Alteon partnership for the first time will bring the no-hop routing capability to non-ATM systems. Officials said IBM plans to extend this capability to Ethernet, Fast Ethernet, Token Ring, and FDDI servers and PCs through future partnerships.

IBM's MSS, available as a stand-alone server or as a module on the IBM 8260 Nways enterprise switch, combines LAN Emulation functions with Layer 3 switching based on the ATM Forum's Multiprotocol Over ATM specification.

Alteon's AceNIC is scheduled to begin volume shipments this month. The company says that with its Gigabit Ethernet AceSwitch, the devices can exchange data in fewer, larger packets than those used in traditional Ethernet, reducing the number of interruptions to a server's CPU when large data transfers are taking place.

Analysts said the partnership, and others like it, will provide a much-needed link between ATM and other technologies.

"With this approach, even if today you don't have hard-core ATM applications, you can exploit ATM's high-speed switching," said Atul Kapoor, president of Kaptronics, a consulting company in Haworth, N.J.

Alteon officials said adapters with MSS capability will be available by the end of the year. Pricing has not been set.

IBM's Networking Hardware Division, in Raleigh, N.C., is at (800) 426-2255 or http://www.networking.ibm.com. Alteon Networks Inc., in San Jose, Calif., is at (408) 574-5500 or http://www.alteon.com.

IBM's ATM route server reaches out to Gigabit Ethernet

A server equipped with a Gigabit Ethernet adapter from Alteon will be able to act as a Multiprotocol Switched Services (MSS) client, receiving and caching route information from an MSS server so it can send data directly to its destination on a different network segment

1. First packet makes one hop before going on to the end-user. 2. MSS shares route information. 3. Subsequent packets traverse the network with no hops.

SEAT Mark Klein outlines his vision for IBM's new Enterprise Integration Business Unit

Ties that bind

THE CLEAREST message at IBM's e-business day in San Francisco recently was the need to combine Lotus' weight in the collaboration market with IBM's strength in enterprise systems and databases. At the crux of that effort is the newly created Enterprise Integration Business Unit and its General Manager Mark Klein, the former Lotus director of product development at Edge Research. *InfoWorld* reporter Amy Dean recently caught up with Klein to discuss his new role.

What's the new business unit's charter?

It's to interconnect disparate servers and to make the data and information on them accessible by a variety of clients. We're going to deliver solutions data as well as applications. There's only been piecemeal work in this space until now. We don't see this as a niche business at all. It's very big, very broad-based, and something we've been doing less formally for a number of years. Microsoft is just beginning to roll out products, and Netscape hasn't got anything but white papers.

Who are your competitors and how do you differentiate yourself from them?

There are two classes. Most obviously, there's Netscape and Microsoft. Netscape is saying that they're getting stuff that we already do very well — objects store and replication. Microsoft is a little further ahead. But besides that, there are a group of the smaller, more focused competitors specific to the business

> KLEIN page 52

CISCO 3800 TO EASE WAN CONNECTIONS

By Stephen Lawson

CISCO INTRODUCED last week at NetWorld+Interop in Las Vegas a series of access concentrators that aggregate voice, video, LAN, and legacy traffic onto a frame-relay or ATM service provider network.

The Cisco 3800 platform offers multiservice frame-relay connectivity and can be upgraded to Asynchronous Transfer Mode (ATM) with a software upgrade. It will allow enterprises to exchange all types of data between remote offices and central sites via the same link. Providers can use a 3800 to offer data, voice, and video service over frame-relay networks and future ATM backbones with the same device.

A queuing feature can separate real-time voice and video traffic from data for low latency and more

> CISCO page 52

INTEL EX.1220.006

28. Alteon was one of only a few known developers of Gigabit networking technology in 1997. A POSA would have been motivated to look to documentation provided by Alteon as a reference. Several large corporations had partnered with Alteon to promote its Gigabit Ethernet Network Interface Card. Alteon and Network Appliance demonstrated access to a NetApp F540 filer equipped with an Alteon PCI-bus Gigabit Ethernet Network Interface Card (NIC) at Networld+Interop in 1996: Ex. 1246. Also, Alteon and Sun Microsystems partnered to deliver Gigabit Ethernet products and demonstrated these at Networld+Interop in May 1997. Ex. 1247.

Ex. 1223.014 (Horst Reply Decl.) at ¶ 28;
Ex. 1220.006 (Networking Article);
Paper 45 (241 Reply) at 4.

241 Patent: Disputes

3. Alteon is Prior Art

a) Alteon was available on Alteon.com before the priority date

b) Alteon and Alteon.com were known to POSAs

c) **Patent Owner Submitted a Substantively Identical version of Alteon as Prior Art**

PO submitted a substantively identical version of Alteon as prior art

Ex. 1033 ("Alteon")

Gigabit Ethernet Technical Brief

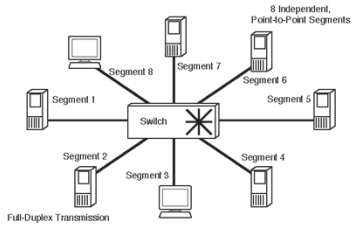


Figure 10. Full-duplex, point-to-point Ethernet segments

An optional flow control mechanism being defined by IEEE 802.3x, is available for full-duplex transmission and works in a way similar to XON/XOFF flow control. A receiving station at one end of the point-to-point connection can send a packet to the sending station at the opposite end of the connection instructing the sending station to stop sending packets for a specified period of time. The sending station ceases to transmit packets until the period of time has passed, or until it receives a new packet from the receiving station with a time of zero, indicating that it is okay to resume transmission.

Full-duplex Ethernet is being standardized by the IEEE 802.3x committee. The full-duplex standard is not specific to any particular speed for Ethernet. It can be used for Ethernet, Fast Ethernet, and Gigabit Ethernet.

Half-duplex Transmission

Using half-duplex transmission, signals travel in both directions on the wire, but not simultaneously. The original 802.3 standard specifies half-duplex transmission.

To gain access to the network in a half-duplex environment, Ethernet has traditionally employed Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as the standard access method. Using CSMA/CD, a station waits for a clear channel. When it detects clear channel, it begins to send frames onto the wire. If two stations start sending data at the same time, a collision occurs. Each station must detect the collision, abort the transmission, and wait for a random interval of time before attempting to transmit data on the network again.

Half-duplex transmission is most commonly used on shared Ethernet segments. Unlike point-to-point connections, shared segments have two or more stations sharing a single port. Figure 11 shows a half-duplex, shared segment.

ALTEON NETWORKS

13

INTEL Ex.1033.017

Ex. 1221 ("1997 Internet Pages from Alteon.com")

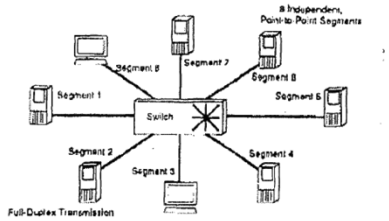


Figure 10. Full-duplex, point-to-point Ethernet segments

An optional flow control mechanism being defined by IEEE 802.3x, is available for full-duplex transmission and works in a way similar to XON/XOFF flow control. A receiving station at one end of the point-to-point connection can send a packet to the sending station at the opposite end of the connection instructing the sending station to stop sending packets for a specified period of time. The sending station ceases to transmit packets until the period of time has passed, or until it receives a new packet from the receiving station with a time of zero, indicating that it is okay to resume transmission.

Full-duplex Ethernet is being standardized by the IEEE 802.3x committee. The full-duplex standard is not specific to any particular speed for Ethernet. It can be used for Ethernet, Fast Ethernet, and Gigabit Ethernet.

Half-duplex Transmission
Using half-duplex transmission, signals travel in both directions on the wire, but not simultaneously. The original 802.3 standard specifies half-duplex transmission.

To gain access to the network in a half-duplex environment, Ethernet has traditionally employed Carrier Sense Multiple Access with Collision Detection (CSMA/CD) as the standard access method. Using CSMA/CD, a station waits for a clear channel. When it detects clear channel, it begins to send frames onto the wire. If two stations start sending data at the same time, a collision occurs. Each station must detect the collision, abort the transmission, and wait for a random interval of time before attempting to transmit data on the network again.

Half-duplex transmission is most commonly used on shared Ethernet segments. Unlike point-to-point connections, shared segments have two or more stations sharing a single port. Figure 11 shows a half-duplex, shared segment.

Intel Ex. 1221.002

Ex. 1033.017 (Alteon); Ex. 1221.002 (PO Submission);
Ex. 1239.017 (Comparison); Ex. 1223.013-.014 (Horst Reply Decl.) at ¶ 27; Paper 45 (241 Reply) at 3-4.

241 Patent: Disputes

4. Motion to Amend 241 Patent should be denied

a) **<u>Patent Owner does not show adequate written description support</u>**

b) Substitute claims 25-32 (Receive) are obvious over Erickson in view of Tanenbaum96 and Alteon

c) Substitute claims 33-48 (Transmit) are obvious over Erickson in view of Tanenbaum96

PO must supply written description support after *Aqua Products*

United States Patent and Trademark Office
Office of the Chief Administrative Patent Judge

MEMORANDUM

TO: Patent Trial and Appeal Board

FROM: David P. Ruschke
Chief Administrative Patent Judge
David.Ruschke@uspto.gov

DATE: November 21, 2017

RE: **Guidance on Motions to Amend in view of *Aqua Products***

Beyond that change, generally speaking, practice and procedure before the Board will not change. For example, a patent owner still must meet the requirements for a motion to amend under 37 C.F.R. § 42.121 or § 42.221, as applicable. That is, a motion to amend must set forth written description support and support for the benefit of a filing date in relation to each substitute claim, and respond to grounds of unpatentability involved in the trial. Likewise, under 37 C.F.R. § 42.11, all parties have a duty of

Paper 54 (Sur-reply for Motion to Amend) at 2.

PO identifies same disclosure for every element without explanation

Claims	Exemplary Support in the '878 Application
Proposed Claim 25	
[[1]]25. A method for network communication, the method comprising:	*See, e.g.*, Ex. 2021 at Abstract, Figs. 3, Figs. 4A, 4B, 4C, and 4D, ¶¶ [0055]-[0064], Cl. 1.
receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;	*See, e.g.*, Ex. 2021 at Abstract, Figs. 3, Figs. 4A, 4B, 4C, and 4D, ¶¶ [0055]-[0064], Cl. 1.
processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;	*See, e.g.*, Ex. 2021 at Abstract, Figs. 3, Figs. 4A, 4B, 4C, and 4D, ¶¶ [0055]-[0064], Cl. 1.
sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;	*See, e.g.*, Ex. 2021 at Abstract, Figs. 3, Figs. 4A, 4B, 4C, and 4D, ¶¶ [0055]-[0064], Cl. 1.
sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application running on a host computer without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination or to a host protocol stack running on the host computer.	*See, e.g.*, Ex. 2021 at Abstract, Figs. 3, Figs. 4A, 4B, 4C, and 4D, ¶¶ [0017], [0055]-[0064], Cl. 1.

Paper 25 (Motion to Amend) at Appendix A, p. i (emphasis added); Paper 40 (Opp. Motion to Amend) at 3.

Too late to provide written description support in reply

- Patent Owner provides alleged "exemplary" written description support **for the first time** in its Reply

> **V. A PERSON OF SKILL IN THE ART WOULD UNDERSTAND THAT THE SUBSTITUTE CLAIMS ARE SUPPORTED BY THE '878 APPLICATION AND THE '809 PROVISIONAL APPLICATION**
>
> 19. In my opinion, the '878 Application and '809 Provisional Application provide adequate written description support for the substitute claims.

Paper 46 (Reply ISO Motion to Amend) at 6;
Ex. 2305.006 (Almeroth Decl. ISO Reply) at 6.

Written description support inadequate

- Patent Owner's identified support for sending data to a "*destination in memory allocated to an application running on the host computer*" is insufficient
  - The cited portions of the 878 Application (Ex. 2021) contains no reference to the destination being allocated **<u>to an application</u>**

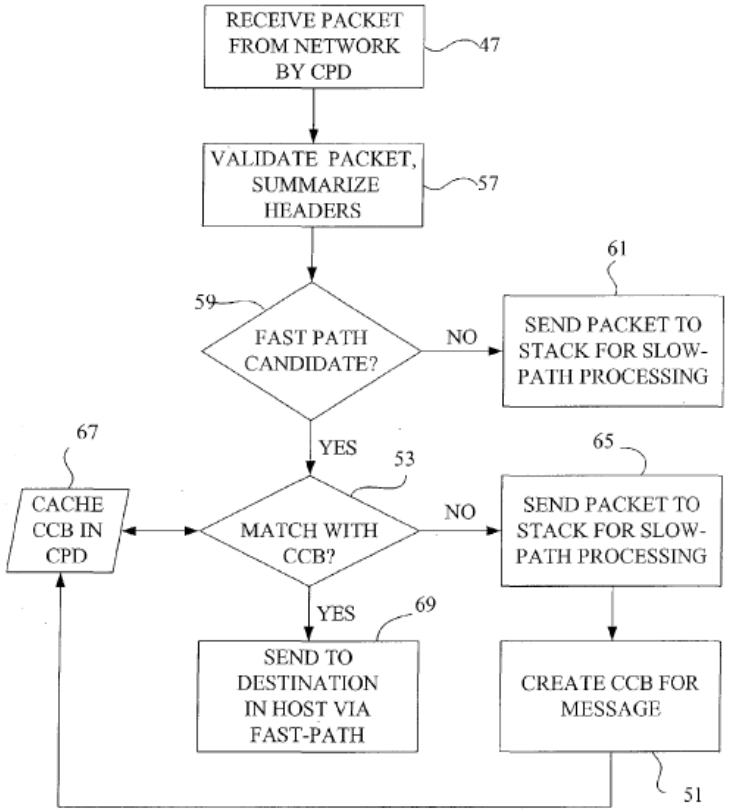


FIG. 3

Ex. 2021 (241 Application) at Fig. 3;
Paper 54 (SurReply ISO Opp Motion to Amend) at 4-5;
Paper 42 (Reply ISO Motion to Amend) at 3.

241 Patent: Disputes

4. Motion to Amend 241 Patent should be denied

a) Patent Owner does not show adequate written description support

b) **<u>Substitute claims 25-32 (Receive) are obvious over Erickson in view of Tanenbaum96 and Alteon</u>**

c) Substitute claims 33-48 (Transmit) are obvious over Erickson in view of Tanenbaum96

New limitation requires that the headers are not sent to a host protocol stack

[[1]]25. A method for network communication, the method comprising:
receiving a plurality of packets from the network, each of the packets including a media access control layer header, a network layer header and a transport layer header;
processing the packets by a first mechanism, so that for each packet the network layer header and the transport layer header are validated without an interrupt dividing the processing of the network layer header and the transport layer header;
sorting the packets, dependent upon the processing, into first and second types of packets, so that the packets of the first type each contain data;
sending, by the first mechanism, the data from each packet of the first type to a destination in memory allocated to an application running on a host computer without sending any of the media access control layer headers, network layer headers or transport layer headers to the destination or to a host protocol stack running on the host computer.

Paper 25 (Motion to Amend) at Appendix A, p. i (emphasis added).

Erickson: Transfer of data without headers to the application

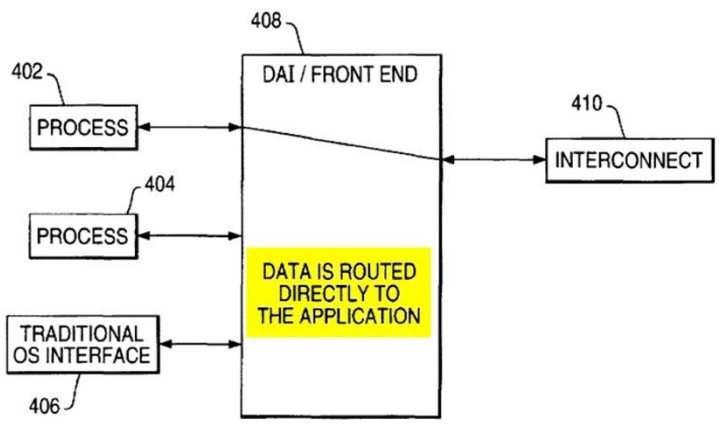


FIG. 4 is a block diagram describing a direct application interface (DAI) and routing of data between processes and an external data connection which is compatible with the present invention. Processes **402** and **404** transmit and receive information directly to and from an interconnect **410** (e.g., I/O device adapter) through the DAI interface **408**. The information coming from the interconnect **410** is routed directly to a process **402** or **404** by use of virtual hardware and registers, rather than using a traditional operating system interface **406**.

Ex. 1005.005, .011 (Erickson) at Fig. 4; 5:6-14;
Paper 4 (Petition) at 57.

241 Patent: Disputes

4. Motion to Amend 241 Patent should be denied

a) Patent Owner does not show adequate written description support

b) Substitute claims 25-32 (Receive) are obvious over Erickson in view of Tanenbaum96 and Alteon

c) **Substitute claims 33-48 (Transmit) are obvious over Erickson in view of Tanenbaum96**

New limitation requires dividing, prepending, and transmitting without an interrupt

[[9]]33. A method for communicating information over a network, the method comprising:	[[17]]41. A method for communicating information over a network, the method comprising:
obtaining data from a source in memory allocated by a first processor;	providing, by a first mechanism, a block of data and a Transmission Control Protocol (TCP) connection;
dividing the data into multiple segments;	dividing, by a second mechanism, the block of data into multiple segments;
prepending a packet header to each of the segments by a second processor, thereby forming a packet corresponding to each segment, each packet header containing a media access control layer header, a network layer header and a transport layer header, wherein the network layer header is Internet Protocol (IP), the transport layer header is Transmission Control Protocol (TCP) and the media access control layer header, the network layer header and the transport layer header are prepended at one time as a sequence of bits during the prepending of each packet header; and transmitting the packets to the network, wherein the dividing, prepending, and transmitting occur without the second processor generating an interrupt to the first processor.	prepending, by the second mechanism, an outbound packet header to each of the segments, thereby forming an outbound packet corresponding to each segment, the outbound packet header containing an outbound media access control layer header, an outbound Internet Protocol (IP) header and an outbound TCP header, wherein the prepending of each outbound packet header occurs without an interrupt dividing the prepending of the outbound media access control layer header, the outbound (IP) header and the outbound TCP header; and
	transmitting the outbound packets to the network, wherein the dividing, prepending, and transmitting occur without the second mechanism generating an interrupt to the first mechanism.

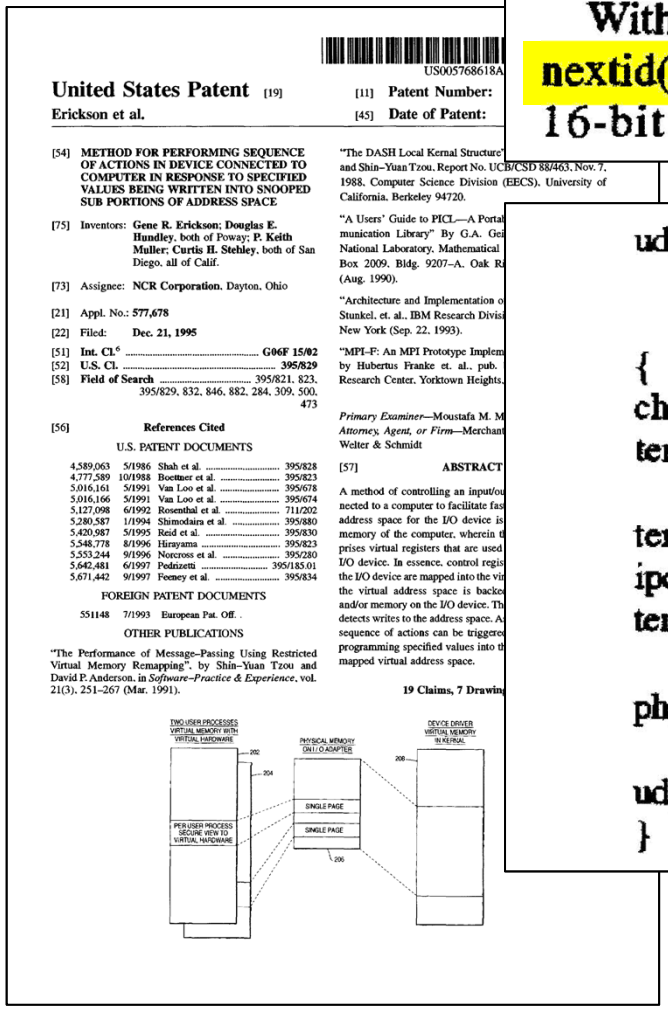
Paper 25 (Motion to Amend) at Appendix A, p. x, xii (emphasis added).

There is “no reason to interrupt the processing of the host computer”

> Regarding the “without interrupt” requirement of claim 17, all processing to generate headers for packets to be sent from the network interface device of Erickson is performed by the processing capability of Erickson’s network interface device with no reason to interrupt the processing of the host computer requesting the transmission.

Paper 11 (Institution Decision) at 19;
Paper 45 (241 Reply) at 14;
Ex. 1223.016 (Horst Reply Decl.) at ¶ 31.

The nextid() function does not require interrupts



US005768618A

United States Patent [19]

Erickson et al.

[11] **Patent Number:**

[45] **Date of Patent:**

[54] **METHOD FOR PERFORMING SEQUENCE OF ACTIONS IN DEVICE CONNECTED TO COMPUTER IN RESPONSE TO SPECIFIED VALUES BEING WRITTEN INTO SNOOPED SUB PORTIONS OF ADDRESS SPACE**

[75] Inventors: **Gene R. Erickson; Douglas E. Hundley**, both of Poway; **P. Keith Muller; Curtis H. Stehley**, both of San Diego, all of Calif.

[73] Assignee: **NCR Corporation**, Dayton, Ohio

[21] Appl. No.: **577,678**

[22] Filed: **Dec. 21, 1995**

[51] **Int. Cl.**[6] **G06F 15/02**

[52] **U.S. Cl.** **395/829**

[58] **Field of Search** 395/821, 823, 395/829, 832, 846, 882, 284, 309, 500, 473

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,589,063	5/1986	Shah et al.	395/828
4,777,589	10/1988	Boettner et al.	395/823
5,016,161	5/1991	Van Loo et al.	395/678
5,016,166	5/1991	Van Loo et al.	395/674
5,127,098	6/1992	Rosenthal et al.	711/202
5,280,587	1/1994	Shimodaira et al.	395/880
5,420,987	5/1995	Reid et al.	395/830
5,548,778	8/1996	Hirayama	395/823
5,553,244	9/1996	Norcross et al.	395/280
5,642,481	6/1997	Pedrizetti	395/185.01
5,671,442	9/1997	Feeney et al.	395/834

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message–Passing Using Restricted Virtual Memory Remapping", by Shin–Yuan Tzou and David P. Anderson, in *Software–Practice & Experience*, vol. 21(3), 251–267 (Mar. 1991).

"The DASH Local Kernal Structure" and Shin–Yuan Tzou, Report No. UCB/CSD 88/463, Nov. 7, 1988, Computer Science Division (EECS), University of California, Berkeley 94720.

"A Users' Guide to PICL—A Portal munication Library" By G.A. Gei National Laboratory, Mathematical Box 2009, Bldg. 9207–A, Oak Ri (Aug. 1990).

"Architecture and Implementation o Stunkel, et. al., IBM Research Divisi New York (Sep. 22, 1993).

"MPI–F: An MPI Prototype Implem by Hubertus Franke et. al., pub. Research Center, Yorktown Heights,

Primary Examiner—Moustafa M. M *Attorney, Agent, or Firm*—Merchant Welter & Schmidt

[57] **ABSTRACT**

A method of controlling an input/ou nected to a computer to facilitate fas address space for the I/O device is memory of the computer, wherein t prises virtual registers that are used I/O device. In essence, control regis the I/O device are mapped into the vir the virtual address space is backe and/or memory on the I/O device. Th detects writes to the address space. A sequence of actions can be triggere programming specified values into th mapped virtual address space.

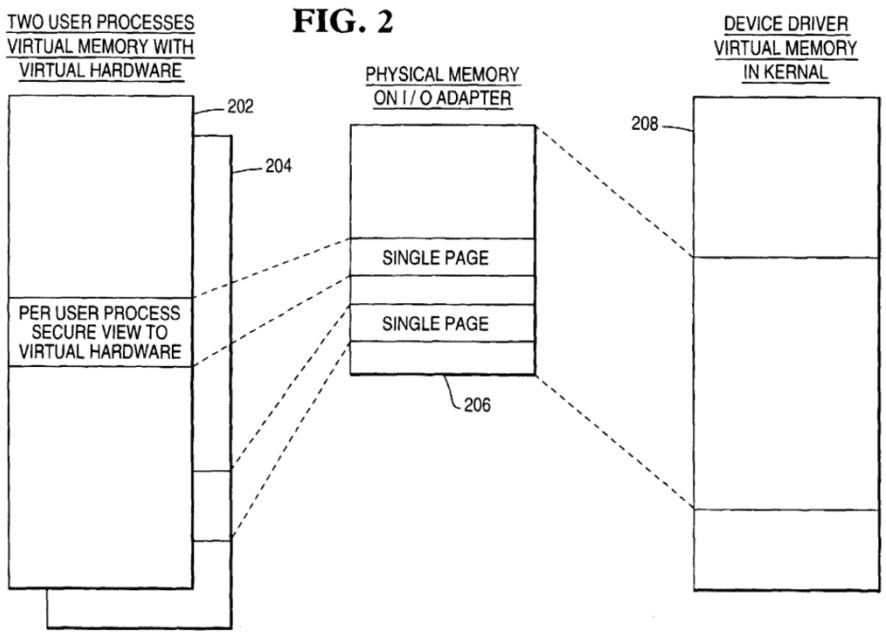
19 Claims, 7 Drawing

Within the udpscript procedure described above, the nextid( ) function provides a monotonically increasing 16-bit counter required by the IP protocol. The

```
udpscript (void *USERDATA_ADDRESS,
          int        USERDATA_LENGTH,
                     template_t *template)
{
char *physaddress;
template->IP.TotalLength = sizeof (IPHeader) +
          sizeof(UDPHeader) + USERDATA_LENGTH;
template->IP.DatagramID = nextid() ;
ipchecksum (template) ;
template->UDPLength = sizeof (UDPHeader)
          + USERDATA_LENGTH;
physaddress = vtophys (USERDATA_ADDRESS,
          USERDATA_LENGTH) ;
udpchecksum (physaddress, USERDATA_LENGTH, template) ;
}
```

Ex. 1005.012 (Erickson) at 7:50-65, 8:10-12;
Ex. 1255 (Horst Amend SurReply Decl.) at ¶¶ 12-14;
Paper 54 (SurReply Motion to Amend) at 10-11.

Erickson teaches the transfer to data without interrupts via polling



"Incoming data is then written to the virtual memory and detected by polling or "snooping" hardware. The snooping hardware, after detecting the write to virtual registers, generates an exception for the system bus controller."

Ex. 1005.012 (Erickson) at 8:56-57;
Ex. 1223.025-.026 (Horst Reply Decl.) at ¶¶ 50-52;
Paper 45 (241 Reply) at 6;
Paper 54 (SurReply ISO Mtn. to Amend) at 11.