Intel Corp. v. Alacritech, Inc.

IPR2017-01391, -01392, -01393, -01405, -01406, -01409, -01410

September 13, 2018

Demonstratives: Table of Contents

Demonstratives: Table of Contents

Demonstratives: Table of Contents

Demonstratives: Table of Contents

(7) Common Issues (Continued)

U.S. Patent No. 7,237,036 (036 Patent)

IPR2017-1391 (Intel)
IPR2018-0371 (Dell)
IPR2017-1718 (Cavium)
IPR2018-0327 (Wistron)

*All citations herein are to the IPR2017-01391 case unless otherwise noted.

036 Patent: Instituted Grounds

- **Erickson** in view of **Tanenbaum96**
  - 036 Patent: Claims 1, 2-7

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

036 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 1-7 of the 036 Patent

3. Motion to Amend 036 Patent should be denied

036 Patent: Disputes

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

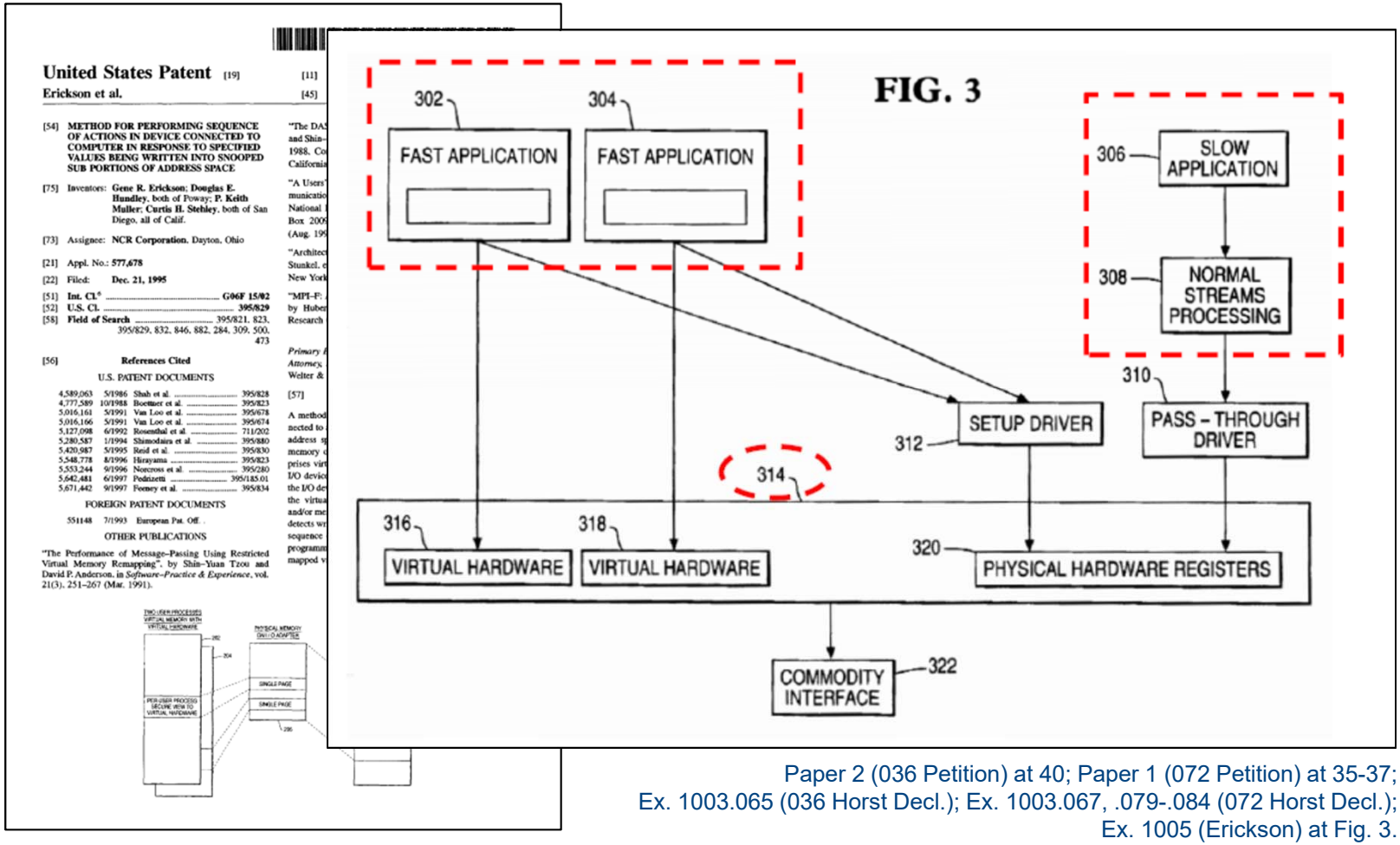
   a) **A POSA would have naturally looked to Tanenbaum96 when implementing the TCP functionality disclosed in Erickson**

   b) Tanenbaum96 does not teach away from the invention

   c) A POSA would have a reasonable expectation of success using Tanenbaum96 to implement Erickson's TCP functionality

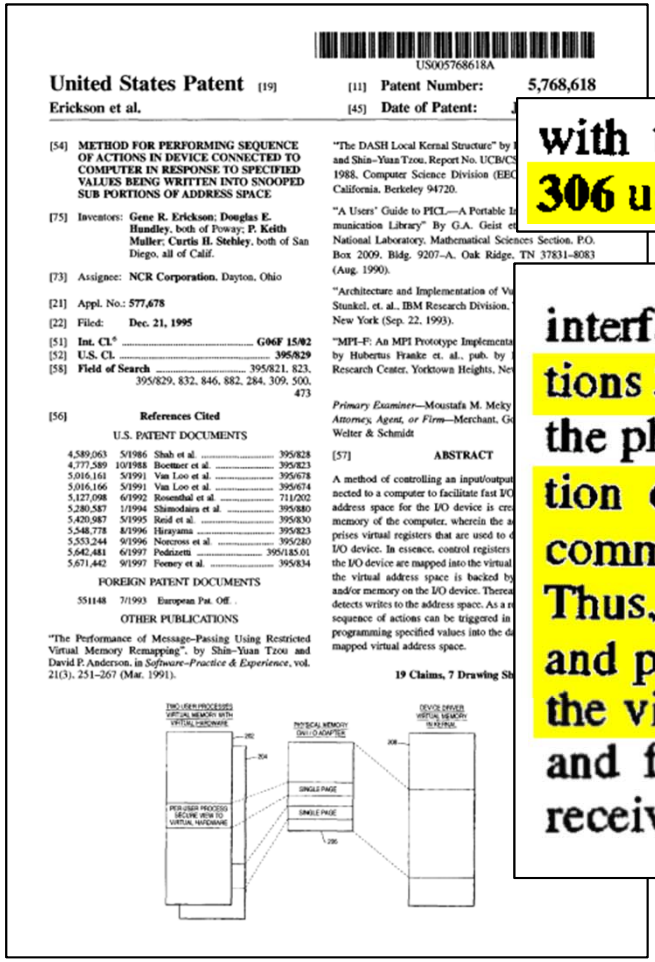
   d) Dr. Horst's 2001 Article shows that "conventional wisdom" was to offload TCP

Erickson: Use of fast and slow applications



Paper 2 (036 Petition) at 40; Paper 1 (072 Petition) at 35-37;
Ex. 1003.065 (036 Horst Decl.); Ex. 1003.067, .079-.084 (072 Horst Decl.);
Ex. 1005 (Erickson) at Fig. 3.

Erickson: Adapter offloads protocol processing for fast applications



US005768618A

United States Patent [19]

Erickson et al.

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

[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" by I and Shin-Yuan Tzou, Report No. UCB/CS 1988, Computer Science Division (EEC California, Berkeley 94720.

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

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

"MPI-F: An MPI Prototype Implementa by Hubertus Franke et. al., pub. by I Research Center, Yorktown Heights, Ne

Primary Examiner—Moustafa M. Meky *Attorney, Agent, or Firm*—Merchant, Go 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 cre memory of the computer, wherein the a prises virtual registers that are used to c I/O device. In essence, control registers the I/O device are mapped into the virtual the virtual address space is backed by and/or memory on the I/O device. Therea detects writes to the address space. As a r sequence of actions can be triggered in programming specified values into the da mapped virtual address space.

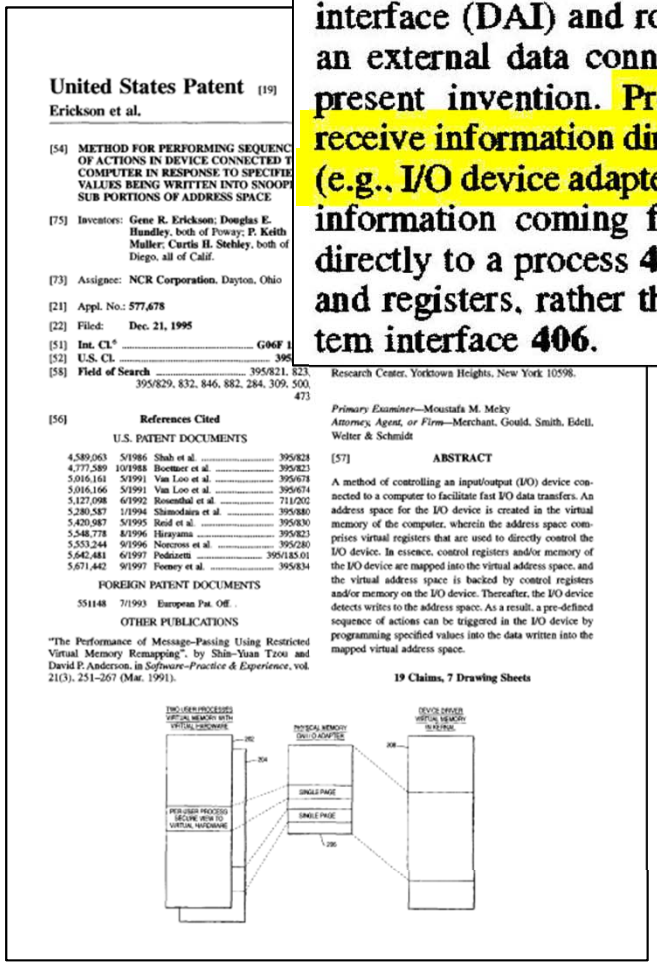
19 Claims, 7 Drawing Sh

with the present invention. A traditional slow application **306** uses normal streams processing **308** to send information

interface **322**. With the present invention, fast user applications **302** and **304** directly use a setup driver **312** to initialize the physical hardware registers **320**, then send the information directly through the I/O device adapter **314** to the commodity interface **322** via virtual hardware **316** and **318**. Thus, the overhead of the normal streams processing **308** and pass-through driver **310** are eliminated with the use of the virtual hardware **316** and **318** of the present invention, and fast applications **302** and **304** are able to send and receive information more quickly than slow application **306**.

Paper 2 (036 Petition) at 40-41; Paper 1 (072 Petition) at 35-37; Ex. 1003.065-.066 (036 Horst Decl.); Ex. 1003.067-.068, .079-.084 (072 Horst Decl.); Ex. 1005 (Erickson) at 4:53-5:3.

Erickson: Fast receive and transmit



United States Patent [19]
Erickson et al.

[54] METHOD FOR PERFORMING SEQUENC OF ACTIONS IN DEVICE CONNECTED T COMPUTER IN RESPONSE TO SPECIFIE VALUES BEING WRITTEN INTO SNOOP SUB PORTIONS OF ADDRESS SPACE

[75] Inventors: Gene R. Erickson; Douglas E. Hundley, both of Poway; P. Keith Muller; Curtis H. Stehley, both of 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 1

[52] U.S. Cl. 395

[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).

Research Center, Yorktown Heights, New York 10598.

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

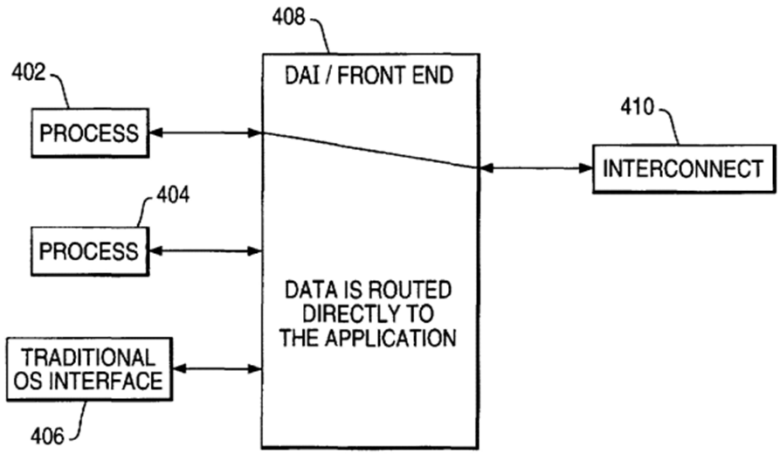
[57] ABSTRACT

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

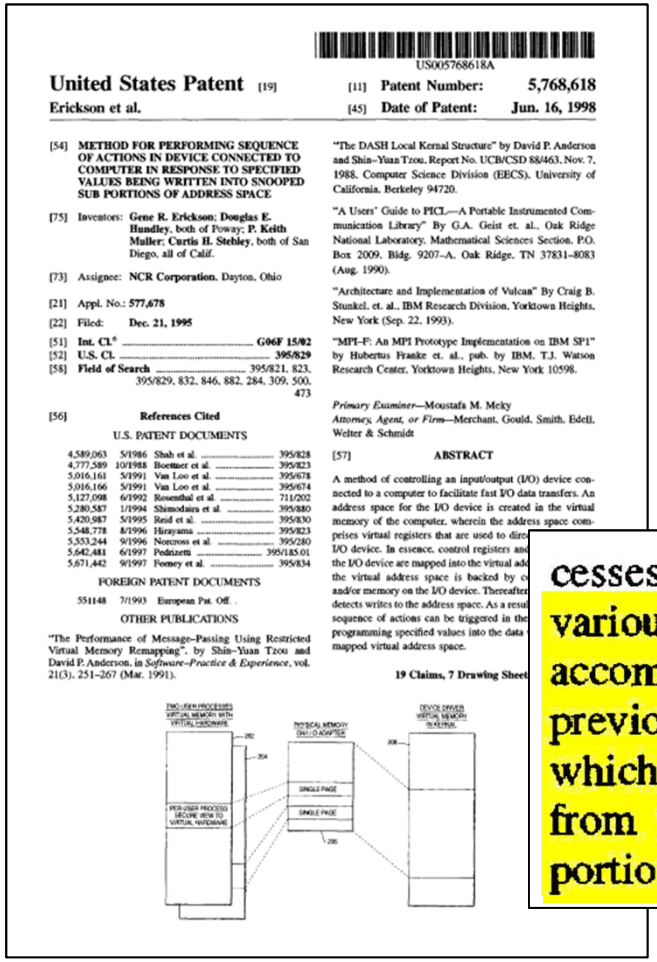
FIG. 4 is a block diagram describing a direct application interface (DAI) and routing of data between processes and an external data connection which is compatible with the present invention. Processes **402** and **404** transmit and receive information directly to and from an interconnect **410** (e.g., 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**.

FIG. 4



Paper 2 (036 Petition) at 44-45; Paper 1 (072 Petition) at 35-37;
Ex. 1003.077, .079-.084 (072 Horst Decl.); Ex. 1005 (Erickson) at 5:6-14, Fig. 4.

Erickson: Adapter stores protocol scripts and data for moving 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	Pedzierzi	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).

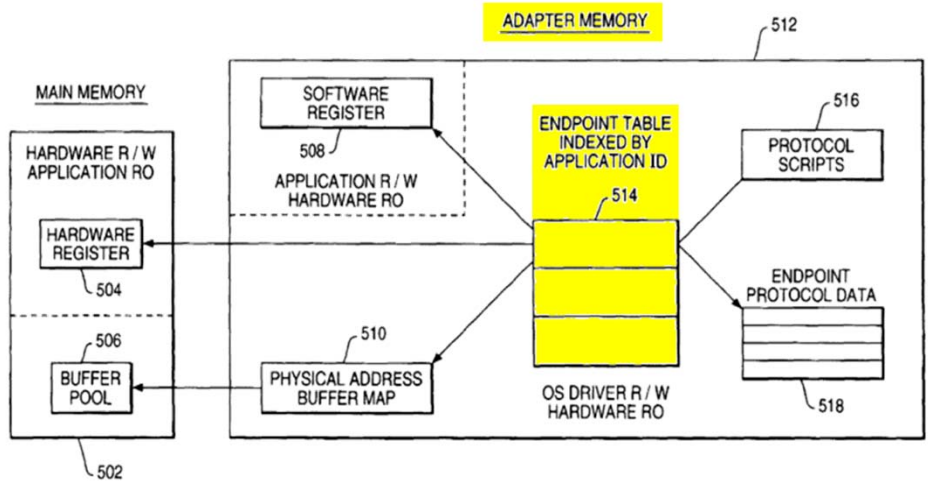
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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

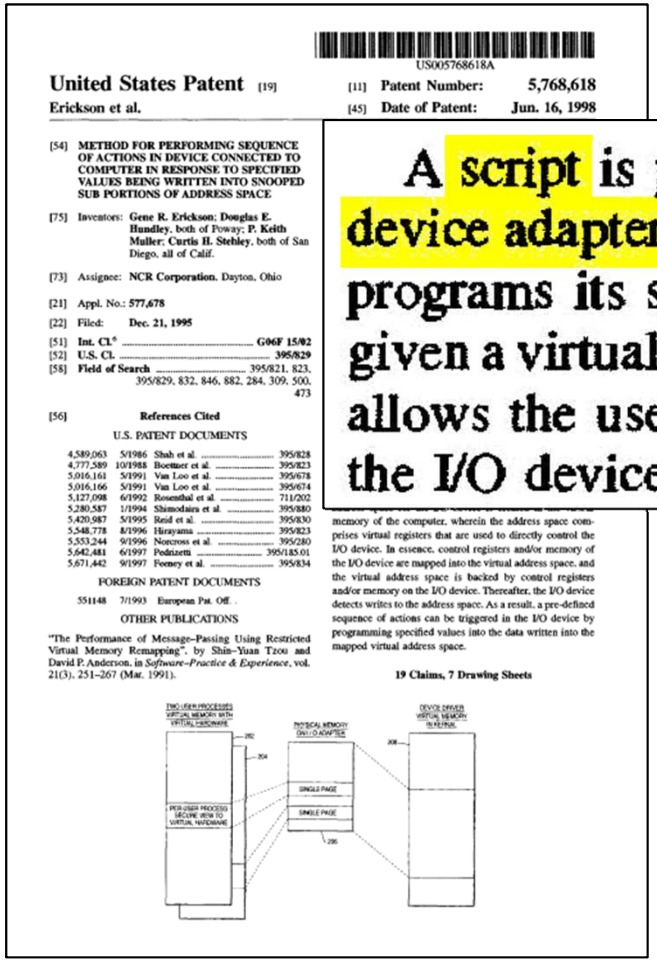
19 Claims, 7 Drawing Sheet



cesses. Each entry within the endpoint table **514** points to various protocol data **518** in the memory **512** in order to accommodate multiple communication protocols, as well as previously defined protocol scripts **516** in the memory **512**, which indicate how data or information is to be transferred from the memory **512** of the I/O device adapter to the portions of main memory **502** associated with a user process.

Paper 2 (036 Petition) at 41-42; Paper 1 (072 Petition) at 37;
Ex. 1005 (Erickson) at 5:61-67.

Erickson: Adapter executes the scripts



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).

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

A script is prepared by the operating system for the I/O device adapter to execute each time the specific user process programs its specific virtual hardware. The user process is given a virtual address in the user process' address space that allows the user process very specific access capabilities to the I/O device adapter.

Paper 2 (036 Petition) at 65; Paper 41 (036 Reply) at 15;
Paper 1 (072 Petition) at 40-41; Ex. 1003.094 (072 Horst Decl.); Ex. 1005 (Erickson) at 4:18-23.

Erickson: A pre-negotiated template passed to the script on the adapter

FIG. 7

Hex	Dec				
0	0	Ethernet Header (14 bytes) 704	01	Target Ethernet Address (6 bytes)	702
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) 706	45	Version = 4, IP Header Len	
f	15		00	Service Type	
10	16			Total Length	
11	17				
12	18			Datagram Id	
13	19				
14	20		40	Flag 0x4 DO_NOT_FRAGM	
15	21		00	Fragment Offset = 0x000	
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 = 128	
1b	27		01		
1c	28		c0		
1d	29		07		
1e	30		80	IP Address of Destination =	
1f	31		01		
20	32		c0		
21	33		08		
22	34	UDP Header (8 bytes) 708	00	Source Port = 0x0007 (ech	
23	35		07		
24	36		30	Destination Port = 0x3018	
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 2 (036 Petition) at 45, 56, 65-66; Paper 1 (072 Petition) at 41-42, 53; Ex. 1005 (Erickson) at 7:65-8:9, Fig. 7; Ex. 1003.096-.097, .111 (072 Horst Decl.); Ex. 1003.093-.094, .104 -.105 (036 Horst Decl.).

Erickson: “Pre-negotiated” header template includes “almost everything”

FIG. 6

602

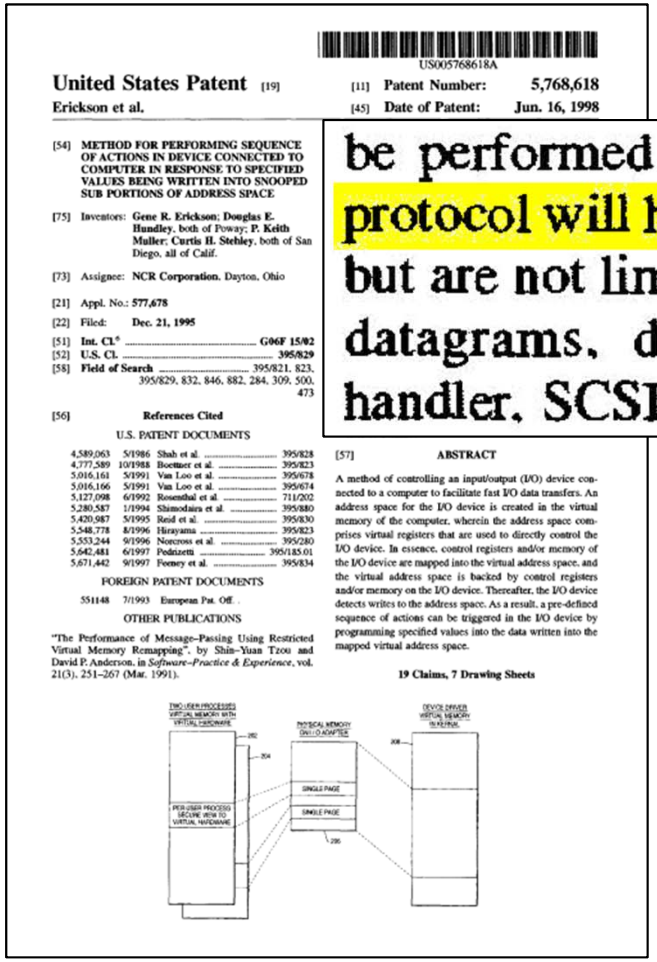
Hex	Dec			
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)
1	1		02	
2	2		03	
3	3		04	
4	4		05	
5	5		06	
6	6		07	Source Ethernet Address (6 bytes)
7	7		08	
8	8		09	
9	9		0a	
a	10		0b	
b	11		0c	
c	12		08	Protocol Type (0x0800 = IP)
d	13		00	
e	14	IP Header (20 bytes) 606	45	Version = 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		00	
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 2 (036 Petition) at 42, 56-57; Paper 1 (072 Petition) at 40, 53, 63, Ex. 1003.093, -.095 (036 Horst Decl.); Ex. 1003.095 -.096, .112, (072 Horst Decl.); Ex. 1005 (Erickson) at 6:57-7:4, Figure 6.

Erickson: Adapter uses scripts for multiple protocols including 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	Neacross et al.	395/280
5,642,481	6/1997	Podzimek	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 2 (036 Petition) at 43, 46, 58; Paper 41 (036 Reply) at 2; Paper 1 (072 Petition) at 42, 44, 47, 53; Paper 46 (072 Reply) at 2; Ex. 1003.095, .107, .120 (036 Horst Decl.); Ex. 1003.093, .096, .101 (072 Horst Decl.); Ex. 1005 (Erickson) at 5:41-51.

Erickson: Identifies Tanenbaum as a reference for TCP

nicate with each other. A discussion of the form and structure of TCP sockets and packets, which are well-known within the art, may be found in many references, including *Computer Networks* by Andrew S. Tanenbaum, Prentice-Hall, New Jersey, 1981, pp. 326–327, 373–377, which is herein incorporated by reference.

Paper 2 (036 Petition) at 46;
Paper 1 (072 Petition) at 34;
Ex. 1005 (Erickson) at 4:37-44.

A POSA following Erickson's suggestion would consult the then-current (1996) edition of Tanenbaum to implement Erickson's TCP script

Paper 2 (036 Petition) at 46; Paper 41 (036 Reply) at 5;
Paper 1 (072 Petition) at 35; Paper 46 (072 Reply) at 5;
Ex. 1003.077 (036 Horst Decl.) ¶ 139;
Ex. 1003.079 (072 Horst Decl.) ¶ 138.

PO's expert taught Tanenbaum96 before alleged priority date Oct. 1997

Fall 1997

Andrew S. Tanenbaum, Computer Networks, 3rd Edition, Prentice-Hall, 1996.

5/1/2018

CS 176 -- Introduc

Fall 1997

Course In

Lecture: Tu

Discussion: Fri

Monday 12:00pm to 12:50pm (Broida 1015)

WWW Page:

http://www.cs.ucsb.edu/~cs176/

Textbook:

Andrew S. Tanenbaum, Computer Networks, 3rd Edition, Prentice-Hall, 1996.

Required Prerequisites:

CS 130A-B -- Data Structures and Algorithms I-II

Helpful Prerequisites:

PSTAT 120A -- Probability and Statistics

Professor Information

Kevin Almeroth
Office 2113, Engineering I
E-mail: almeroth@cs.ucsb.edu
Phone: 893-2777
Office Hours: Tuesday/Thursday; 1:45pm - 2:45pm (and open door policy)

Teaching Assistant Information

Hongjun Zhu
Office: CSIL, 1138, Engineering I
E-mail: hongjunz@cs.ucsb.edu
Office Hours: Monday, 2:00pm-4:00pm; Thursday, 3:00pm-5:00pm or by appointment.

Student Evaluation

Ext. bit 21
Almeroth
Thurman, CSR 8834

http://www.cs.ucsb.edu/~almeroth/classes/F97.176/syllabus.html

INTEL EX.1234.001

1/3

Q. Dr. Almeroth, do you recognize Exhibit 21?

A. It looks like the front page for the first course at UCSB that I taught.

Q. And the textbook was the Tanenbaum '96, right, that's the basis for several of the grounds that we've been talking about today and yesterday, right?

A. Yes. It was

Paper 41 (036 Reply) at 10; Paper 46 (072 Reply) at 10; Paper 60 (036 Opp. to Motion to Exclude) at 5-6; Paper 64 (072 Opp. to Motion to Exclude) at 5-6; Ex. 1234 (Almeroth Dep., Ex. 21); Ex. 1225.219 (Almeroth Depo.) at 474:21-475:2.

PO patents describe Tanenbaum96 as a college-level textbook

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

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

(75) Inventors: **Laurence B. Boucher**, Saratoga, C (US); **Stephen E. J. Blightman**, Sa Jose, CA (US); **Peter K. Craft**, Sa Francisco, CA (US); **David A. Hig** Saratoga, CA (US); **Clive M. Phill** 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 o patent is extended or adjusted und U.S.C. 154(b) by 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52

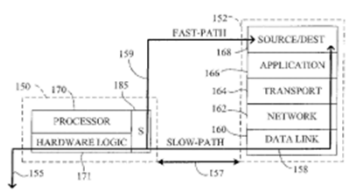
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

tion processing device (CPD) associated with a host computer. 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.

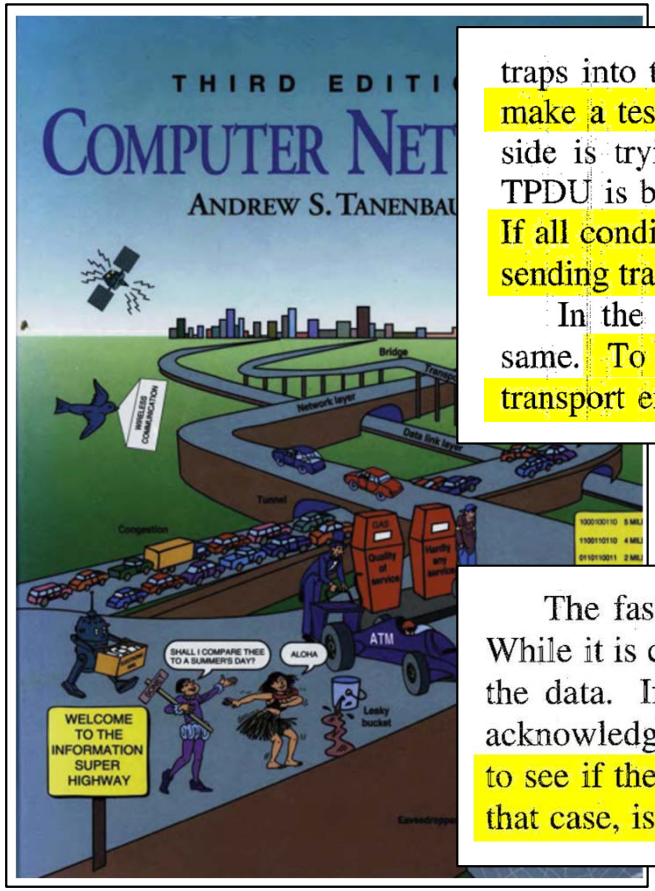
22 Claims, 89 Drawing Sheets



The above description of layered protocol processing is simplified, as college-level textbooks devoted primarily to this subject are available, such as Computer Networks, Third Edition (1996) by Andrew S. Tanenbaum, which is incor-

Paper 60 (036 Opp. to Mot. to Exclude) at 13;
Paper 64 (072 Opp. to Mot. to Exclude) at 13;
Ex. 1001 (036 Patent) at 4:47-50;
Ex. 1001 (072 Patent) at 4:57-60;
See also Paper 2 (036 Petition) at 34; Paper 41 (036 Reply) at 10;
Paper 1 (072 Petition) at 28; Paper 46 (072 Reply) at 10.

Tanenbaum96: “Fast path” processing using a prototype header



traps into the kernel to do the SEND. The first thing the transport entity does is make a test to see if this is the normal case: the state is *ESTABLISHED*, neither side is trying to close the connection, a regular (i.e., not an out-of-band) full TPDU is being sent, and there is enough window space available at the receiver. If all conditions are met, no further tests are needed and the fast path through the sending transport entity can be taken.

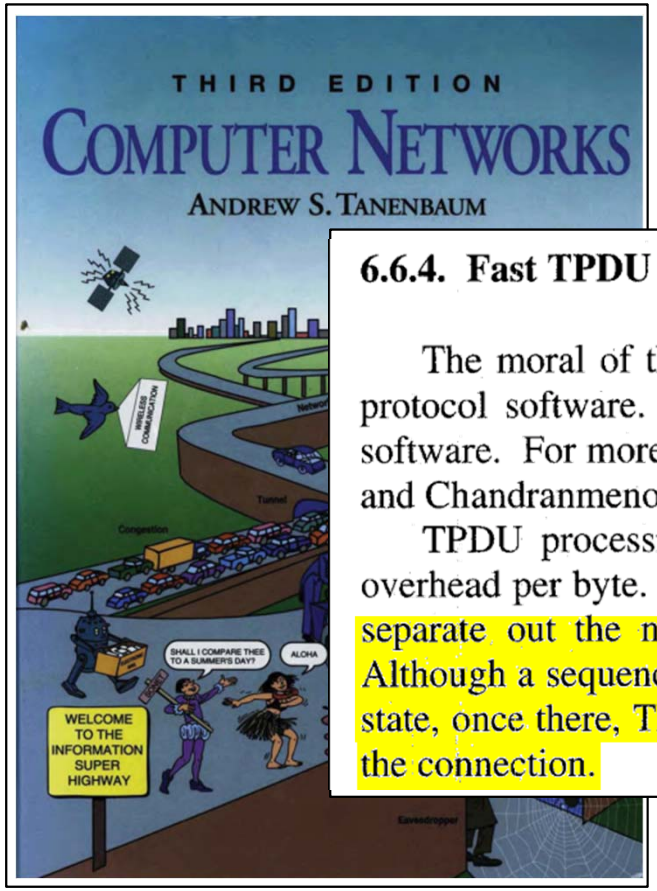
In the normal case, the headers of consecutive data TPDUs are almost the same. To take advantage of this fact, a prototype header is stored within the transport entity. At the start of the fast path, it is copied as fast as possible to a

Paper 2 (036 Petition) at 35, 47-49; Ex.1003.059, .079-.083 (036 Horst Decl.);
Paper 1 (072 Petition) at 28-29,35-38; Ex. 1003.061,.079-.084 (072 Horst Decl.);
Ex. 1006.583 (Tanenbaum96).

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 2 (036 Petition) at 37, 47-49; Ex.1003.062, -079-.083 (036 Horst Decl.);
Paper 1 (072 Petition) at 30,35-38; Ex. 1003.064,-.079-.084 (072 Horst Decl.);
Ex. 1006.585 (Tanenbaum96).

Tanenbaum96: Protocol processing is "straightforward" for the "normal case"



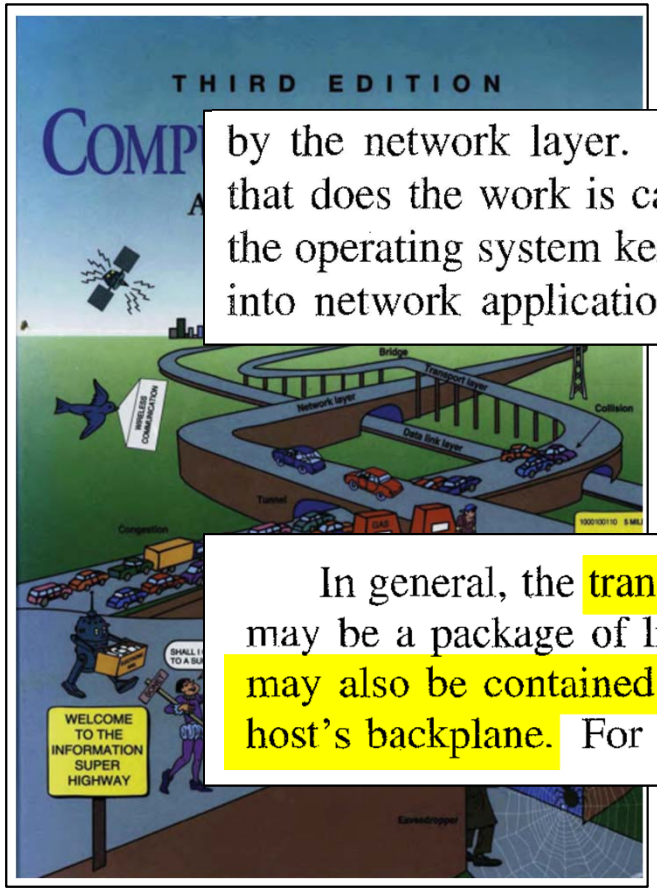
6.6.4. Fast TPDU Processing

The moral of the story above is that the main obstacle to fast networking is protocol software. In this section we will look at some ways to speed up this software. For more information, see (Clark et al., 1989; Edwards and Muir, 1995; and Chandranmenon and Varghese, 1995).

TPDU processing overhead has two components: overhead per TPDU and overhead per byte. Both must be attacked. The key to fast TPDU processing is to separate out the normal case (one-way data transfer) and handle it specially. Although a sequence of special TPDUs are needed to get into the *ESTABLISHED* state, once there, TPDU processing is straightforward until one side starts to close the connection.

Paper 2 (036 Petition) at 15, 49, 58, 62; Paper 41 (036 Reply) at 7-8;
Paper 1 (072 Petition) at 14-15, 28-29, 37-39; Paper (072 Reply) at 7-8;
Ex.1003.033, .082, .096, .100 (036 Horst Decl.);
Ex. 1003.034, .084 (072 Horst Decl.);
Ex. 1006.583 (Tanenbaum96).

Tanenbaum96: Transport entity may reside on network interface



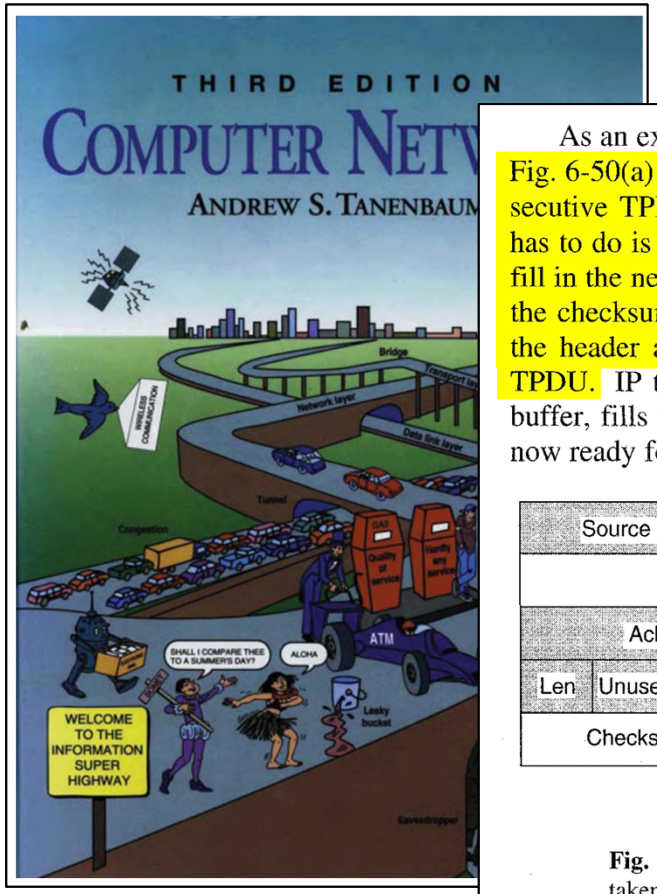
by the network layer. The hardware and/or software within the transport layer that does the work is called the **transport entity**. The transport entity can be in the operating system kernel, in a separate user process, in a library package bound into network applications, or on the network interface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

Paper 2 (036 Petition) at 35-36, 47; Paper 41 (036 Reply) at 6;
Paper 1 (072 Petition) at 44; Paper 46 (072 Reply) at 6;
Ex. 1003.059, .064 (036 Horst Decl.); Ex. 1003.062, .066, (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 41 (036 Reply) at 6;
Paper 46 (072 Reply) at 6;
Ex. 1006.530 (Tanenbaum96).

Fast path transmit reuses the prototype header



As an example of how this principle works in practice, let us consider TCP/IP. Fig. 6-50(a) shows the TCP header. The fields that are the same between consecutive TPDUs on a one-way flow are shaded. All the sending transport entity has to do is copy the five words from the prototype header into the output buffer, fill in the next sequence number (by copying it from a word in memory), compute the checksum, and increment the sequence number in memory. It can then hand the header and data to a special IP procedure for sending a regular, maximum TPDU. IP then copies its five-word prototype header [see Fig. 6-50(b)] into the buffer, fills in the *Identification* field, and computes its checksum. The packet is now ready for transmission.

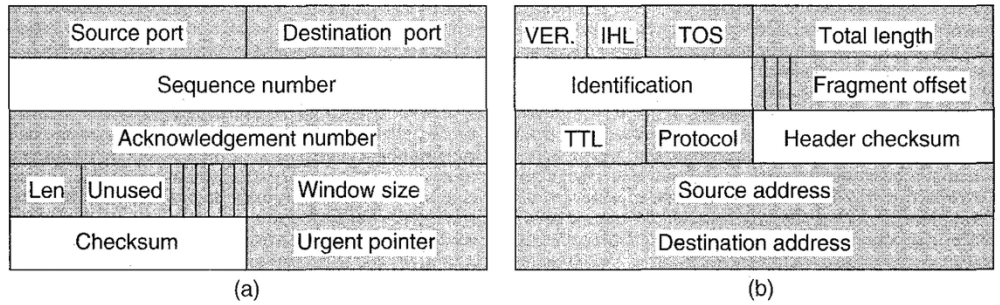


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 2 (036 Petition) at 36, 47-49; Paper 1 (072 Petition) at 29, 35-37;
Ex. 1003.061, .077-082 (036 Horst Decl.); Ex. 1003.063, .080-.085 (072 Horst Decl.);
Ex. 1006.584 (Tanenbaum96).

Tanenbaum96 teaches how to modify Erickson's template header for TCP

FIG. 6

Hex	Dec				
0	0	Ethernet Header (14 bytes) 604	01	Target Ethernet Address (6 bytes)	602
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		00		
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"	

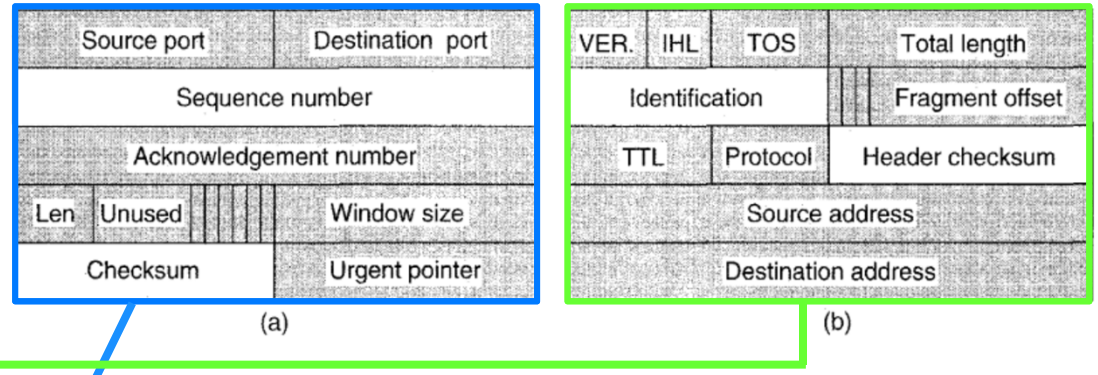
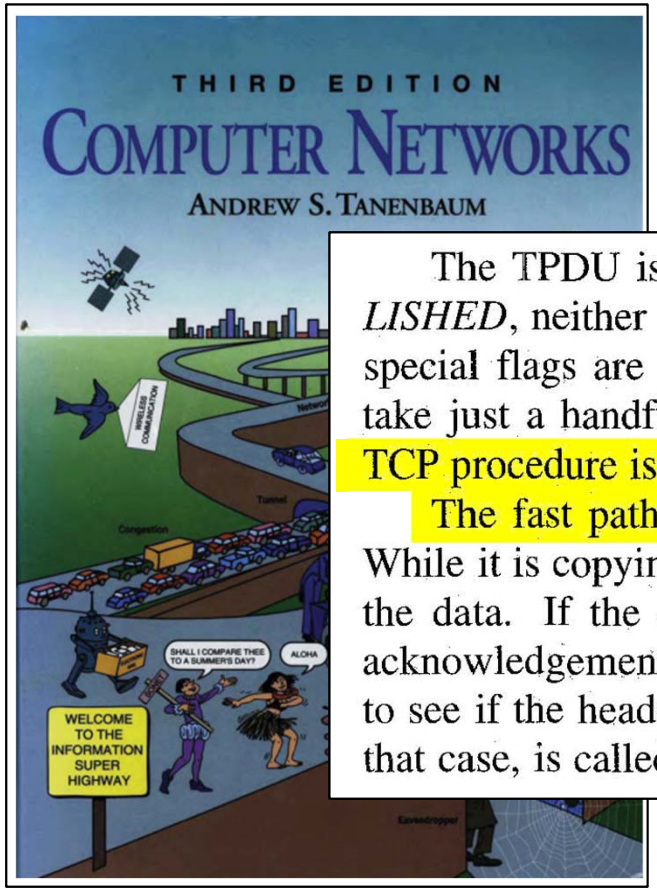


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 2 (036 Petition) at 36, 47-49, 54-63; Paper 1 (072 Petition) at 29, 35-37, 51-55;
Ex. 1003.061, .077-082, .091-.101 (036 Horst Decl.);
Ex. 1003.063, .080-.085, .112-.113 (072 Horst Decl.);
Ex. 1006.584 (Tanenbaum96);
Ex. 1005 (Erickson) Fig. 6.

Fast path receive updates a connection record and copies data to user memory

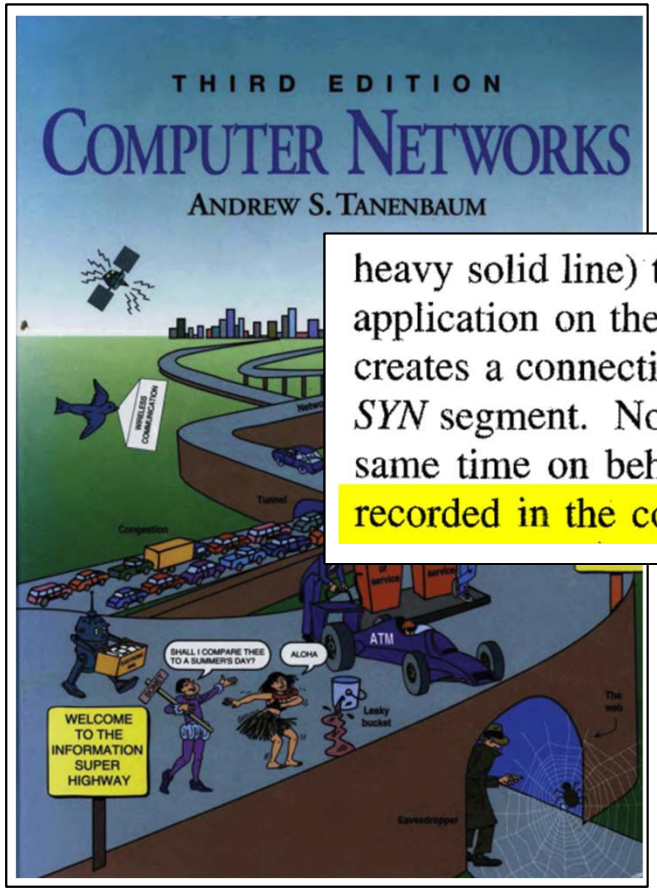


The TPDU is then checked to see if it is a normal one: the state is *ESTABLISHED*, neither side is trying to close the connection, the TPDU is a full one, no special flags are set, and the sequence number is the one expected. These tests take just a handful of instructions. If all conditions are met, a special fast path TCP procedure is called.

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 2 (036 Petition) at 37-39, 48-49, 54-63; Paper 1 (072 Petition) at 30-31, 35-37, 70-71; Ex. 1003.060, .078-082 (036 Horst Decl.); Ex. 1003.064, .080-.085 (072 Horst Decl.); Ex. 1006.585 (Tanenbaum96).

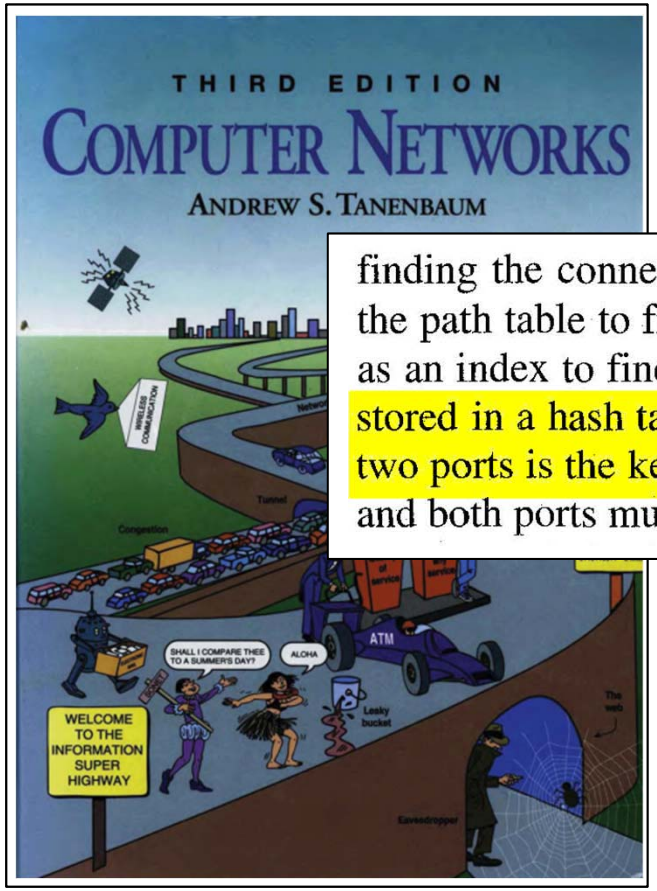
The connection record stores TCP state information



heavy solid line) then later the path of a server (the heavy dashed line). When an application on the client machine issues a CONNECT request, the local TCP entity creates a connection record, marks it as being in the *SYN SENT* state, and sends a *SYN* segment. Note that many connections may be open (or being opened) at the same time on behalf of multiple applications, so the state is per connection and recorded in the connection record. When the *SYN+ACK* arrives, TCP sends the

Paper 2 (036 Petition) at 38, 48-49, 54-63;
Paper 1 (072 Petition) at 30, 35-37;
Ex. 1003.065, .078-082, .091-.101 (036 Horst Decl.);
Ex. 1003.065, .080-.085 (072 Horst Decl.);
Ex. 1006.549 (Tanenbaum96).

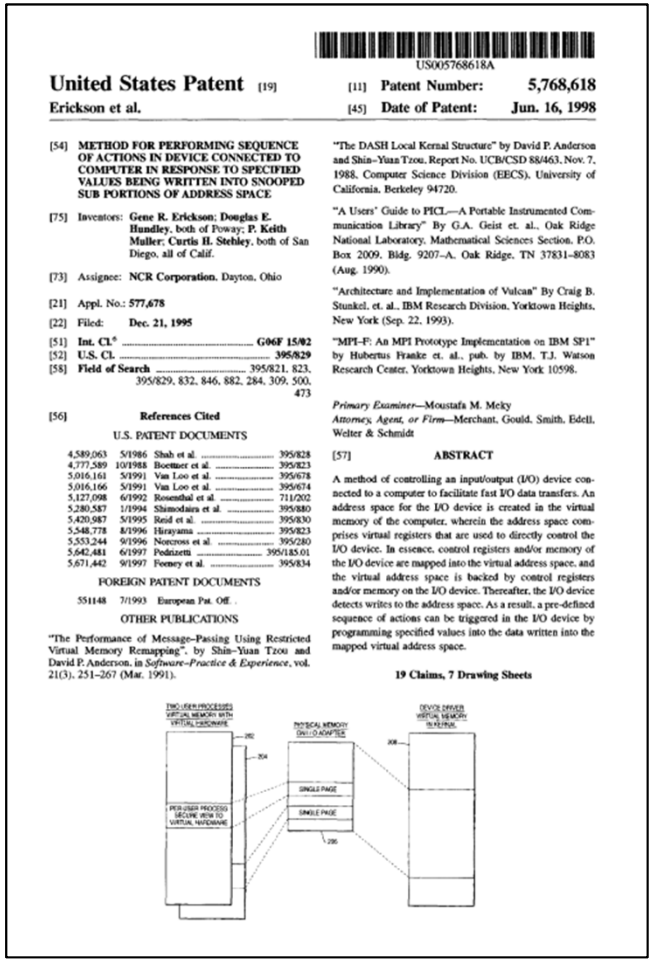
The connection record is looked up using the IP addresses and TCP ports



finding the connection record is easy: the *VPI* field can be used as an index into the path table to find the virtual circuit table for that path and the *VCI* can be used as an index to find the connection record. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses and both ports must be compared to verify that the correct record has been found.

Paper 2 (036 Petition) at 37-38, 47-49, 54-63;
Paper 1 (072 Petition) at 30, 35-37;
Ex. 1003.064-.065, .078-082, .091-.101 (036 Horst Decl.);
Ex. 1003.064-.065, .080-.085 (072 Horst Decl.);
Ex. 1006.585 (Tanenbaum96).

Tanenbaum96 teaches how to modify Erickson's endpoint table for TCP



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).

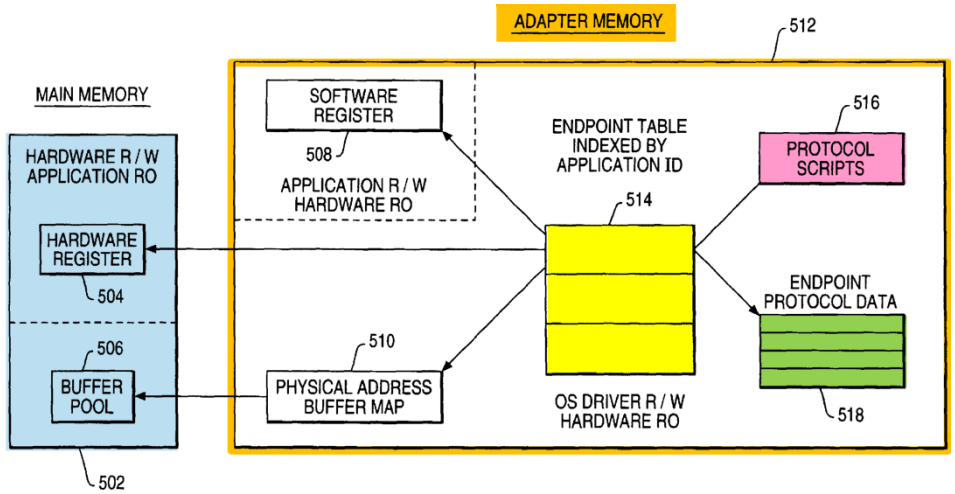
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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



cesses. Each entry within the endpoint table **514** points to various protocol data **518** in the memory **512** in order to accommodate multiple communication protocols, as well as previously defined protocol scripts **516** in the memory **512**, which indicate how data or information is to be transferred from the memory **512** of the I/O device adapter to the portions of main memory **502** associated with a user process.

Paper 2 (036 Petition) at 41-42, 47-49; Paper 1 (072 Petition) at 34-37; Ex. 1003.067-.068, .079-.082 (036 Horst Decl.); Ex. 1003.069-.070, .081-.084 (072 Horst Decl.); Ex. 1005 (Erickson) at 5:53-67, Fig. 5.

Connection record in Tanenbaum96 corresponds to endpoint data in Erickson

Tanenbaum96

as an index to find the connection record. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses

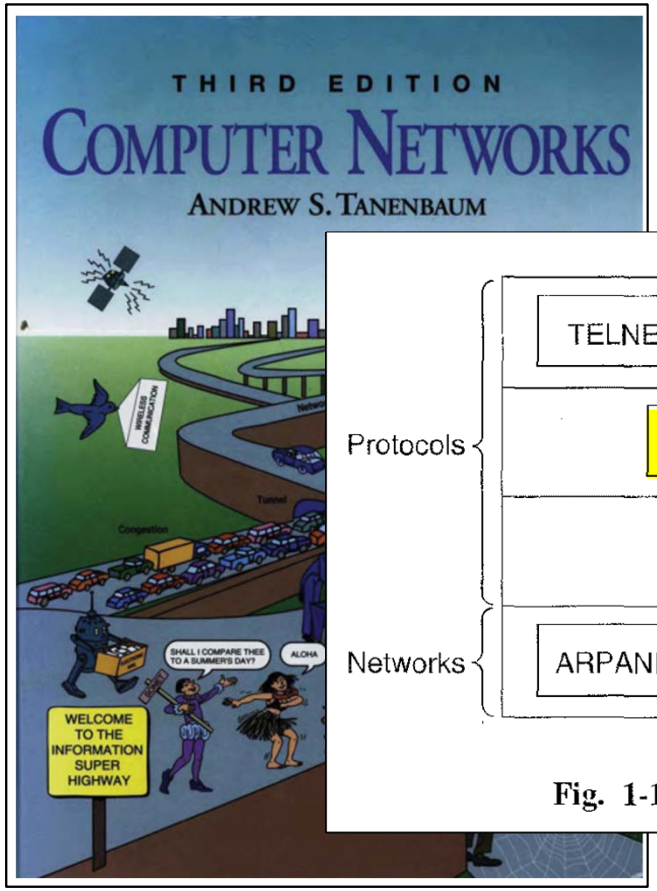
Paper 2 (036 Petition) at 37-38, 47-49, 54-63; Paper 1 (072 Petition) at 30, 35-37; Ex. 1003.064-.065, .078-082, .091-.101 (036 Horst Decl.); Ex. 1003.064-.065, .080-.085 (072 Horst Decl.); Ex. 1006.585 (Tanenbaum96).

Erickson

cesses. Each entry within the endpoint table **514** points to various protocol data **518** in the memory **512** in order to accommodate multiple communication protocols, as well as previously defined protocol scripts **516** in the memory **512**, which indicate how data or information is to be transferred from the memory **512** of the I/O device adapter to the portions of main memory **502** associated with a user process.

Paper 2 (036 Petition) at 41-42, 47-49; Paper 1 (072 Petition) at 34-37; Ex. 1003.067-.068, .079-.082 (036 Horst Decl.); Ex. 1003.069-.070, .081-.084 (072 Horst Decl.); Ex. 1005 (Erickson) at 5:53-67, Fig. 5.

TCP and UDP are alternative protocols for the TCP/IP protocol suite



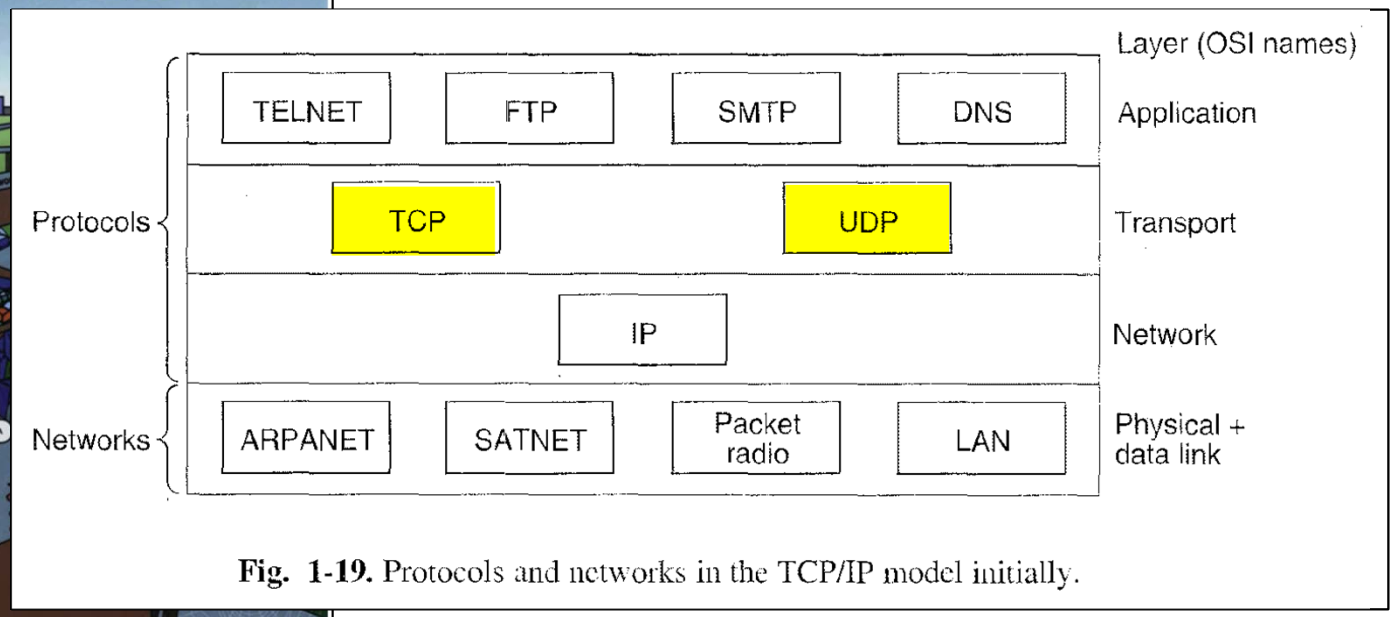
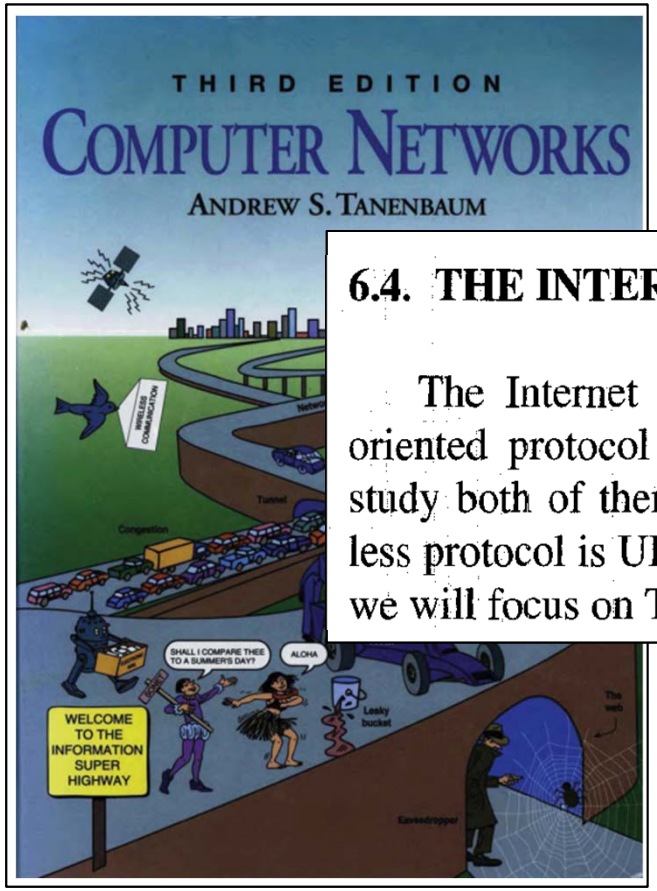


Fig. 1-19. Protocols and networks in the TCP/IP model initially.

Paper 2 (036 Petition) at 21; Paper 41 (036 Reply) at 2;
Paper 1 (072 Petition) at 39; Paper 46 (072 Reply) at 2;
Ex. 1003.060 (072 Horst Decl.); Ex. 1003.057 (036 Horst Decl.);
Ex. 1006.055 (Tanenbaum96).

TCP and UDP: “Two main protocols” for IP



6.4. THE INTERNET TRANSPORT PROTOCOLS (TCP AND UDP)

The Internet has two main protocols in the transport layer, a connection-oriented protocol and a connectionless one. In the following sections we will study both of them. The connection-oriented protocol is TCP. The connection-less protocol is UDP. Because UDP is basically just IP with a short header added, we will focus on TCP.

Paper 2 (036 Petition) at 21; Paper 41 (036 Reply) at 3;
Paper 1 (072 Petition) at 18-19; Paper 46 (072 Reply) at 3;
Ex. 1003.057 (036 Horst Decl.);
Ex. 1003.060 (072 Horst Decl.);
Ex. 1006.539 (Tanenbaum96).

036 Patent: Disputes

1. A POSA would have been motivated to combine Tanenbaum96 with Erickson
   a) A POSA would have naturally looked to Tanenbaum96 when implementing Erickson's TCP functionality
   b) <u>Tanenbaum96 does not teach away from the invention</u>
   c) A POSA would have a reasonable expectation of success using Tanenbaum96 to implement Erickson's TCP functionality
   d) Dr. Horst's 2001 Article shows that "conventional wisdom" was to offload TCP

Tanenbaum96 does not teach away from invention

PO relies on following passage:

> A tempting way to go fast is to build fast network interfaces in hardware. The difficulty with this strategy is that unless the protocol is exceedingly simple, hardware just means a plug-in board with a second CPU and its own program. To avoid having the network coprocessor be as expensive as the main CPU, it is often a slower chip. The consequence of this design is that much of the time the main (fast) CPU is idle waiting for the second (slow) CPU to do the critical work. It is a myth to think that the main CPU has other work to do while waiting. Furthermore, when two general-purpose CPUs communicate, race conditions can occur, so elaborate protocols are needed between the two processors to synchronize them correctly. Usually, the best approach is to make the protocols simple and have the main CPU do the work.

Paper 30 (036 Response) at 24-25;
Paper 34 (072 Response) at 36-37.

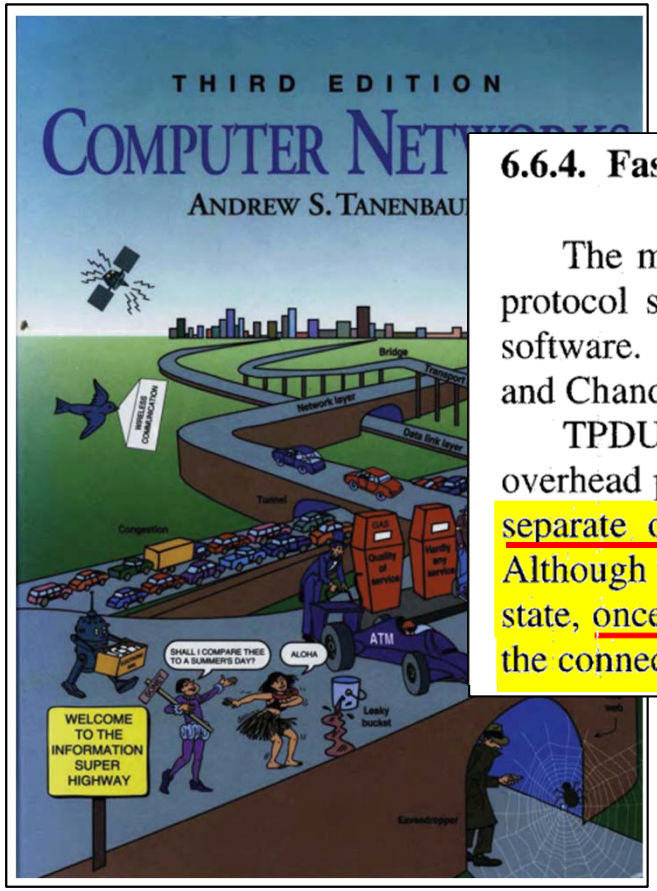
Tanenbaum96 does not teach away from invention

Instead, it describes design preferences and tradeoffs

A tempting way to go fast is to build fast network interfaces in hardware. The difficulty with this strategy is that unless the protocol is exceedingly simple, hardware just means a plug-in board with a second CPU and its own program. To avoid having the network coprocessor be as expensive as the main CPU, it is often a slower chip. The consequence of this design is that much of the time the main (fast) CPU is idle waiting for the second (slow) CPU to do the critical work. It is a myth to think that the main CPU has other work to do while waiting. Furthermore, when two general-purpose CPUs communicate, race conditions can occur, so elaborate protocols are needed between the two processors to synchronize them correctly. Usually, the best approach is to make the protocols simple and have the main CPU do the work.

Paper 42 (036 Reply ISO Motion to Amend) at 6-7;
Paper 46 (072 Reply) at 6-7;
Ex. 1006.588-.589 (Tanenbaum96).

Tanenbaum96: Fast processing is straightforward in the "normal case"



6.6.4. Fast TPDU Processing

The moral of the story above is that the main obstacle to fast networking is protocol software. In this section we will look at some ways to speed up this software. For more information, see (Clark et al., 1989; Edwards and Muir, 1995; and Chandranmenon and Varghese, 1995).

TPDU processing overhead has two components: overhead per TPDU and overhead per byte. Both must be attacked. The key to fast TPDU processing is to separate out the normal case (one-way data transfer) and handle it specially. Although a sequence of special TPDUs are needed to get into the *ESTABLISHED* state, once there, TPDU processing is straightforward until one side starts to close the connection.

Paper 2 (036 Petition) at 15, 49, 58, 62; Paper 41 (036 Reply) at 7-8;
Paper 1 (072 Petition) at 14-15, 28-29, 37-39; Paper (072 Reply) at 7-8;
Ex.1003.033, .082, .096, .100 (036 Horst Decl.);
Ex. 1003.034, .084 (072 Horst Decl.);
Ex. 1006.583 (Tanenbaum96).

PO mischaracterizes the base reference in the combination

PO argues:

> Moreover, Petitioner provides ***no*** explanation as to how, or indeed, why a POSITA would have modified *Tanenbaum* in such a way. Petitioner does not

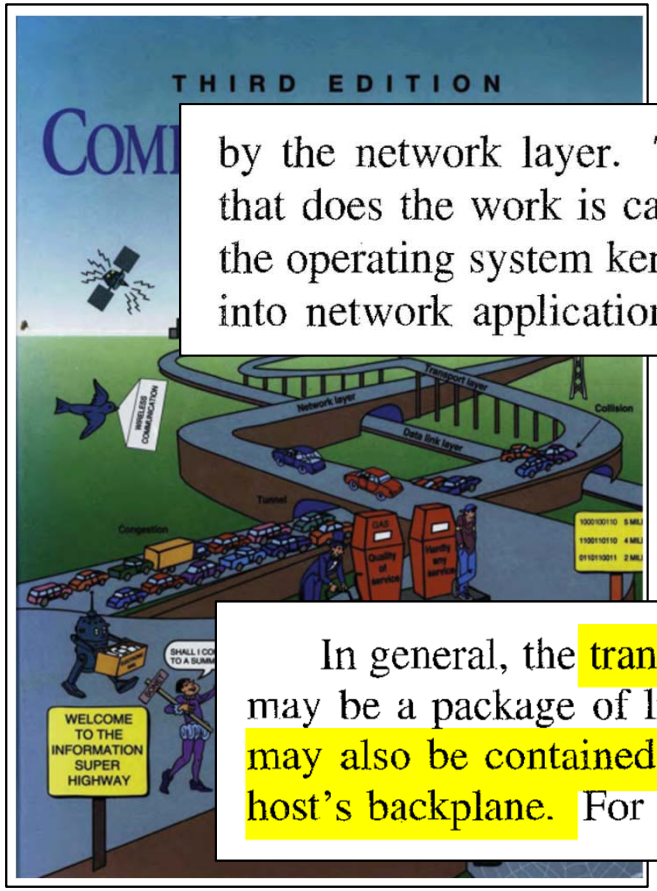
Paper 30 (036 Response) at 26;
Paper 34 (072 Response) at 38.

<u>But</u> Petitioner is relying on modification of Erickson:

> As set forth below, Erickson in view of Tanenbaum96 renders obvious claims 1-7 of the 036 Patent. Erickson discloses the large majority of the limitations through its description of fast path protocol processing of UDP/IP by an I/O device adapter. Tanenbaum96 discloses corresponding details for TCP/IP. In

Paper 2 (036 Petition) at 50; Paper 41 (036 Reply) at 6;
see also Paper 1 (072 Petition) at 38; Paper 46 (072 Reply) at 6.

Tanenbaum96 also teaches offloading transport layer to interface card



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 2 (036 Petition) at 35-36, 47; Paper 41 (036 Reply) at 6;
Paper 1 (072 Petition) at 44; Paper 46 (072 Reply) at 6;
Ex. 1003.059, .064 (036 Horst Decl.); Ex. 1003.062, .066, (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 41 (036 Reply) at 6;
Paper 46 (072 Reply) at 6;
Ex. 1006.530 (Tanenbaum96).

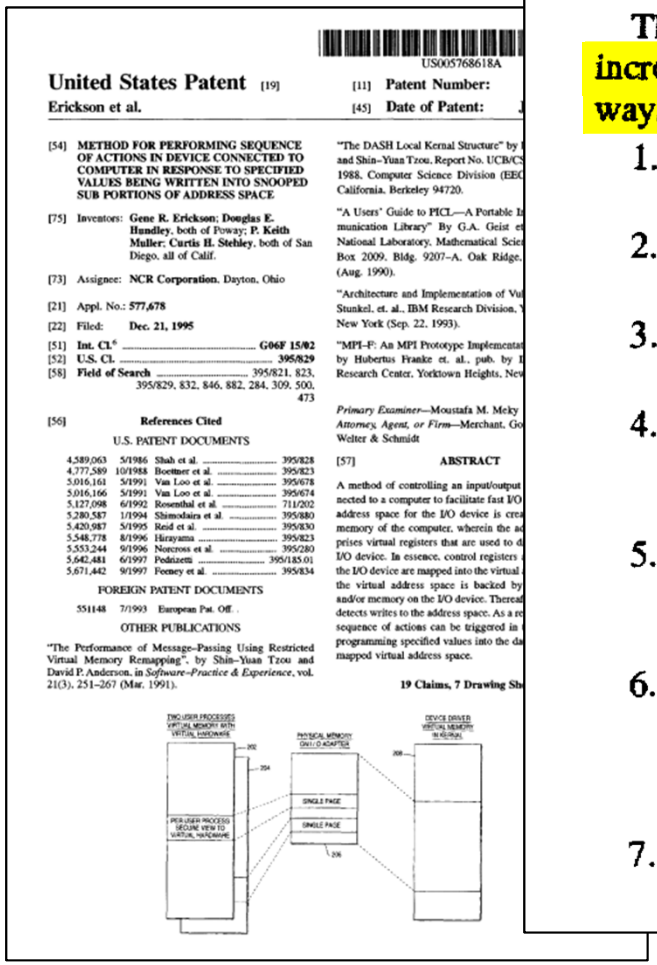
Dr. Horst: Tanenbaum96 does not teach away from the combination

Q. But you wouldn't consider TCP to be an exceedingly simple protocol, would you?

A. The fast path of TCP that's only transferring data is not that complicated. Even the full TCP, there are plenty of examples of people that have solved the problems Tanenbaum is talking about and have done all kinds of different levels of off-loading as I described in my introductory section of the report.

Paper 41 (036 Reply) at 8; Paper 46 (072 Reply) at 7-8;
Ex. 2028 (Horst Dep.) at 135:17-24.

Erickson's benefits apply equally to TCP



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" by
and Shin-Yuan Tzou, Report No. UCB/CS
1988, Computer Science Division (EEC
California, Berkeley 94720.

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

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

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

Primary Examiner—Moustafa M. Meky
Attorney, Agent, or Firm—Merchant, Go
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 crea
memory of the computer, wherein the ad
prises virtual registers that are used to d
I/O device. In essence, control registers
the I/O device are mapped into the virtual
the virtual address space is backed by
and/or memory on the I/O device. Thereaf
detects writes to the address space. As a re
sequence of actions can be triggered in
programming specified values into the da
mapped virtual address space.

19 Claims, 7 Drawing Sh

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

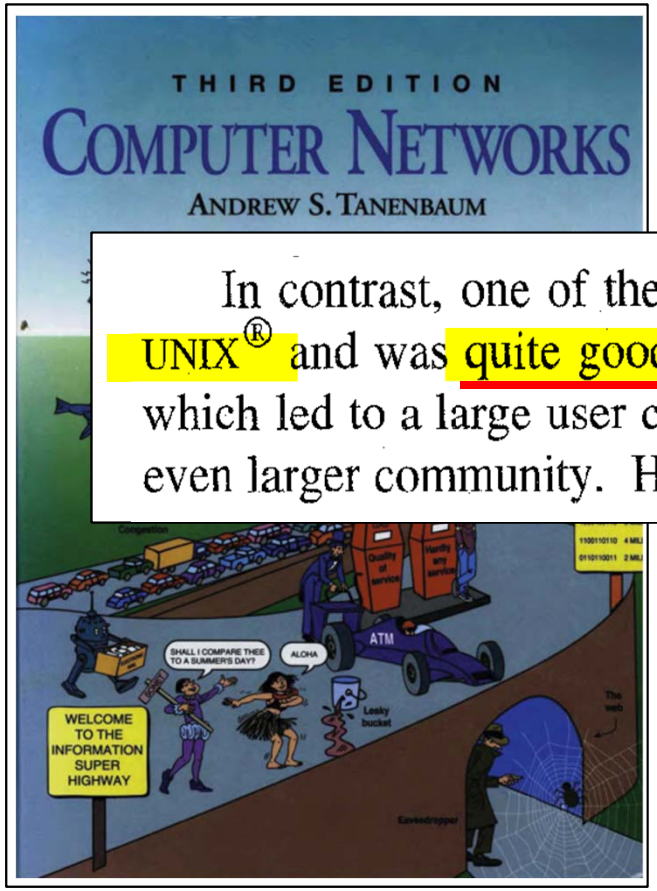
1. Writing information to and from a user address space without intermediate memory-to-memory copies.
2. Accessing an I/O device simultaneously from multiple user processes in a single node.
3. Eliminating calls to the operating system and the associated context switches on a per I/O basis.
4. Maintaining system security for the I/O device by using the operating system to initialize the virtual memory address space of the user process.
5. Accessing the I/O device under the full control of the resource allocation policies and permissions granted by the operating system.
6. Working with a plurality of well-known standard operating systems including, but not limited to, UNIX, OS/2, Microsoft Windows, Microsoft Windows NT, or Novell Netware.
7. Providing low-latency high-performance control of I/O devices.

Paper 41 (036 Reply) at 8; Paper 46 (072 Reply) at 8;
Ex. 1005 (Erickson) at 3:1-22.

036 Patent: Disputes

1. A POSA would have been motivated to combine Tanenbaum96 with Erickson
   a) A POSA would have naturally looked to Tanenbaum96 when implementing Erickson's TCP functionality
   b) Tanenbaum96 does not teach away from the invention
   c) **A POSA would have a reasonable expectation of success using Tanenbaum96 to implement Erickson's TCP functionality**
   d) Dr. Horst's 2001 Article shows that "conventional wisdom" was to offload TCP

Tanenbaum96 identified freely available TCP/IP source code: Berkeley (BSD) UNIX

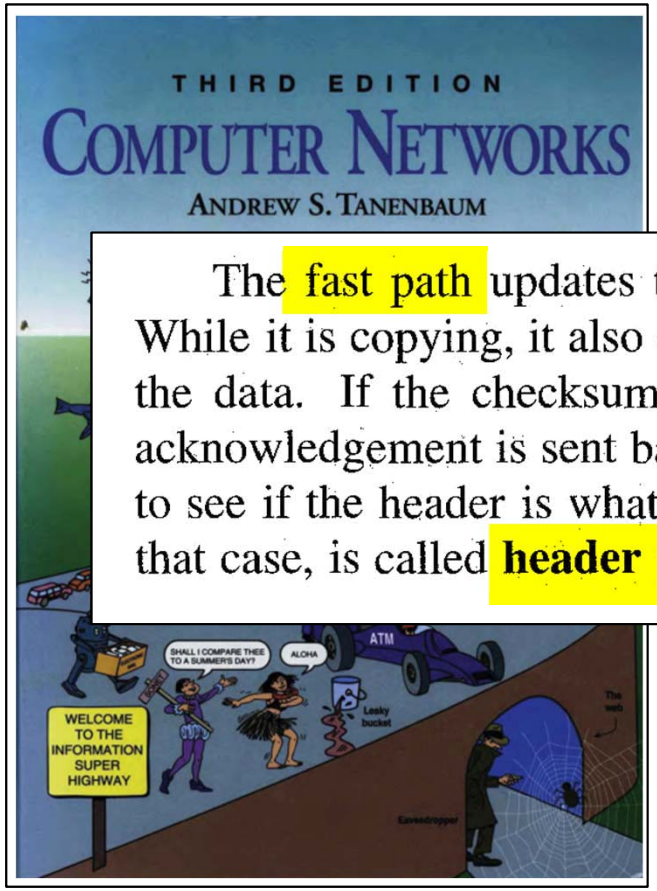


Tanenbaum96 (1996)

In contrast, one of the first implementations of TCP/IP was part of Berkeley UNIX® and was quite good (not to mention, free). People began using it quickly, which led to a large user community, which led to improvements, which led to an even larger community. Here the spiral was upward instead of downward.

Paper 2 (036 Petition) at 15; Paper 41 (036 Reply) at 10;
Paper 1 (072 Petition) at 18; Paper 46 (072 Reply) at 10;
Ex. 1003.013, .020-.021 (036 Horst Decl.) ¶¶ 26, 34; Ex. 1223.011-.014 (036 Horst Reply Decl.) ¶¶ 26-29.
Ex. 1003.014 (072 Horst Decl.) ¶ 26 ; Ex. 1223.011-.014 (072 Horst Reply Decl.) ¶¶ 26-29; Ex. 1006.061 (Tanenbaum96).

Tanenbaum96: Fast path/header prediction is widely used



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 41 (036 Reply) at 10; Paper 46 (072 Reply) at 10;
Ex. 1003.039 (036 Horst Decl.) ¶ 70; Ex. 1003.040 (072 Horst Decl.) ¶ 70;
Ex. 1006.585 (Tanenbaum96).

Berkeley TCP included fast-path header prediction code

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

67. Code to implement the header prediction algorithm was incorporated in the BSD 4.4-Lite distribution.

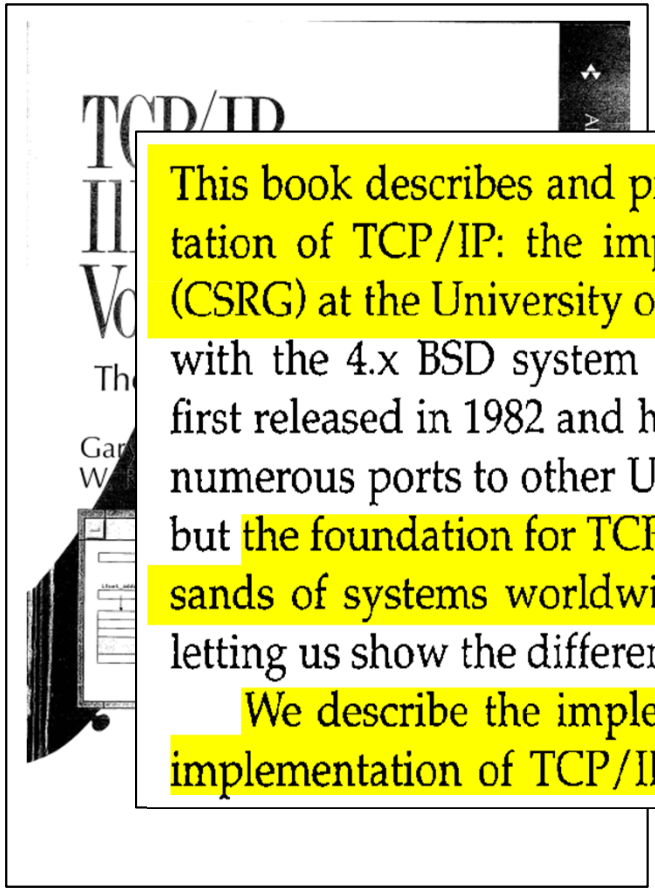
Case IPR. No. **Unassigned**
U.S. Patent No. 7,237,036
Title: FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING A TCP CONNECTION

Declaration of Robert Horst, Ph.D. in Support of Petition for *Inter Partes* Review of U.S. Patent No. 7,237,036

Paper 41 (036 Reply) at 10;
Paper 46 (072 Reply) at 10;
Ex. 1003.037-.038 (036 Horst Decl.) ¶ 67;
Ex. 1003.038-.039 (072 Horst Decl.) ¶ 67.

Other college textbooks documented Berkeley TCP/IP

Stevens Vol. 2 (1995)



This book describes and presents the source code for the common reference implementation of TCP/IP: the implementation from the Computer Systems Research Group (CSRG) at the University of California at Berkeley. Historically this has been distributed with the 4.x BSD system (Berkeley Software Distribution). This implementation was first released in 1982 and has survived many significant changes, much fine tuning, and numerous ports to other Unix and non-Unix systems. This is not a toy implementation, but the foundation for TCP/IP implementations that are run daily on hundreds of thousands of systems worldwide. This implementation also provides router functionality, letting us show the differences between a host implementation of TCP/IP and a router.

We describe the implementation and present the entire source code for the kernel implementation of TCP/IP, approximately 15,000 lines of C code. The version of the

Paper 2 (036 Petition) at 15; Paper 1 (072 Petition) at 18;
Paper 41 (036 Reply) at 10); Paper 46 (072 Reply) at 10;
Ex. 1003.013 (036 Horst Decl.) ¶ 26; Ex. 1003.014-.015 (072 Horst Decl.) ¶ 26;
Ex. 1223.011-.014 (036 Horst Reply Decl.) ¶¶ 26-29; Ex. 1223.011-.014 (072 Horst Reply Decl.) ¶¶ 26-29; Ex.1013.023 (Stevens2).

Stevens2 documented the BSD header prediction code

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Case IPR. No. **Unassigned**
U.S. Patent No. 7,237,036
Title: FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING A TCP CONNECTION

Declaration of Robert Horst, Ph.D. in Support of Petition for *Inter Partes* Review of U.S. Patent No. 7,237,036

Stevens Vol. 2 (1995)

68. The 1995 book (Stevens2) walks through the Jacobson BSD header prediction code including the conditions for selecting the fast or slow path. In order

Paper 41 (036 Reply) at 10;
Paper 46 (072 Reply) at 10;
Ex.1003.038-.039 (036 Horst Decl.) ¶ 68;
Ex. 1003.039-.040 (072 Horst Decl.) ¶ 68.

User data does not need to span page boundaries to support TCP

- PO claims:

> column 8. (Ex. 2026, ¶ 94.) *Erickson* itself acknowledges that its script "would need to be enhanced if the user data were allowed to span page boundaries" (Ex. 1005 at 8:22-24)—something Petitioner's own expert admits is not described by

Paper 30 (036 Response) at 29;
Paper 34 (072 Response) at 41.

- But data does not need to span page boundaries for TCP:

> 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.

Paper 41 (036 Reply) at 12;
Paper 46 (072 Reply) at 11-12);
Ex. 1223.019-.020 (036 Horst Reply Decl.) ¶ 40;
Ex. 1223.019-.020 (072 Horst Reply Decl.) ¶ 40.

Spanning page boundaries would have been straightforward design choice

- Would merely require multiple calls to “vtophys” function disclosed in Erickson

> Header 706 portion of the datagram 702. The vtophys( ) function performs a translation of the user-provided virtual address into a physical address usable by the adapter. In all

Paper 41 (036 Reply) at 12; Paper 46 (072 Reply) at 12;
Ex. 1005 (Erickson) at 8:14-16.

- Erickson contemplates spanning page boundaries

> 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

Paper 41 (036 Reply) at 12; Paper 46 (072 Reply) at 12;
Ex. 1003.032-.033, .046, .049-.050 (036 Horst Decl.) ¶¶ 54,88, 93-94; Ex. 1003.033.034, .047-.048, .051-.052 (072 Horst Decl.) ¶¶ 54,88, 94-95;
Ex. 1223.020 (036 Horst Reply Decl.) ¶¶ 41-43; Ex. 1223.020 (072 Horst Reply Decl.) ¶¶ 41-43;
Ex. 1005 (Erickson) at 8:22-24.

Need design details before you can predict speed

Patent Owner argues:

> During his deposition, Dr. Horst also testified that it is impossible to determine whether executing TCP code on *Erickson*'s adapter would be slower or faster than the traditional slow path:

But speed is unpredictable without design parameters:

> A. Just looking at the code, you couldn't tell. You also need to have information on the compiler, the processor used, the caches. There's all kinds of things that influence the performance of code.

Paper 30 (036 Response) at 40; Paper 34 (072 Response) at 52;
Ex. 2029 (Horst Dep.) at 81:23-82:9;
Paper 41 (036 Reply) at 13; Paper 46 (072 Reply) at 12-13.

Combination does not have to be predictably "faster"

44. Networks are built around standards that specify links with fixed speeds. Over time, the Ethernet physical layer has evolved from 1 mbit/sec, to 10 mbit/sec, to 100 mbit/sec, to Gbit/sec and 10 Gbit/sec. At a given point in time, offloading the networking protocol does not increase the link speed, which must support the then-current standard, but may reduce the processing load on the main CPU and transfer some of the load to the network adapter, freeing the main processor to perform other work. In multiprocessor systems, moving the load between processors is called "load balancing," and the same concept can be applied to networking if the network adapter has sufficient intelligence to take on part of the load. The goal of reducing host processing load has been one of the reasons for much of the prior work on protocol offloads. Guerrero and others have

Paper 41 (036 Reply) at 13; Paper 46 (072 Reply) at 12-13;
Ex. 1223.021-.022 (036 Horst Reply Decl.) ¶ 44; Ex. 1223.021-.022 (072 Horst Reply Decl.) ¶ 44.

Definition of POSA does not matter for obviousness determination

Petitioner:

> and/or networking protocols. PO's Response at 23. While I disagree with this proposed level of ordinary skill, my opinions in this declaration would remain the same even if Patent Owner's opinion concerning the level of ordinary skill in the art were applied.

Ex. 1210.005-.006 (036 Horst Decl. ISO Opp. to Motion to Amend) ¶ 9;
Ex. 1210.005 (072 Horst Decl. ISO Opp. to Motion to Amend) ¶ 9;
Ex. 1223.009 (036 Horst Reply Decl.) ¶ 21;
Ex. 1223.009 (072 Horst Reply Decl.) ¶ 21.

Patent Owner:

> computer networking and/or networking protocols. (Ex. 2026 at ¶ 33.) Any differences between this and Petitioners' proposed level of ordinary skill would have no bearing on the analysis presented in this Response. The cited references

Paper 30 (036 Response) at 22; Paper 34 (072 Response) at 23.

036 Patent: Disputes

1. A POSA would have been motivated to combine Tanenbaum96 with Erickson
   a) A POSA would have naturally looked to Tanenbaum96 when implementing Erickson's TCP functionality
   b) Tanenbaum96 does not teach away from the invention
   c) A POSA would have a reasonable expectation of success using Tanenbaum96 to implement Erickson's TCP functionality
   d) **Dr. Horst's 2001 Article shows that "conventional wisdom" was to offload TCP**

Industry actively working on offloading TCP/IP

IP Storage and the CPU Consumption Myth

Robert Horst
3ware, Inc.
701 E. Middlefield Rd.
Mountain View, CA 94043

Abstract

This paper addresses a key issue t attaching storage devices directly to perceived need for hardware acceleratio networking stack. While many implic acceleration is required, the evidence conclusion is not well founded. In th accelerators have had mixed success, economic justification for hardware acc given the low cost of host CPU cycles. many applications is dominated by a transfer rate, and hardware protocol a little effect on the I/O performance in the Application benchmarks were run on subsystem to measure performance and on Email, database, file serving, and bac The results show that good performance without protocol acceleration.

1. Introduction

The growing popularity of gigab prompted increasing interest in usi networks to attach storage devices to Ethernet Storage Area Networks significant advantages in cost and m compared with Fibre Channel SANs. products are already on the market standardize the protocols is progressing working group of the IETF [1].

Networks customized to storage net Fiber Channel, were developed large perception that standard networking protocols are too heavyweight for attaching storage. Conventional wisdom says that IP storage is impractical without special purpose NICs to accelerate the TCP/IP protocol stack. This papers shows that the need for hardware acceleration is largely a myth. Several different lines of reasoning show that the future of storage networking will rely heavily on storage devices connected to servers without special purpose hardware accelerators.

kills the front-end processor architecture, until the next generation of engineers rediscovers the idea and repeats the cycle.

Some may argue that the problem was that the accelerators should have been optimized hardware instead of embedded programmable processors. Unfortunately, every protocol worthy of acceleration continues to evolve, and it is difficult to stay ahead of the moving target. The new protocols proposed for IP storage, iSCSI and iFCP, are far from stable, and even after the standards have been formally approved, there will likely be a long series of enhancements and bug fixes. It seems extremely

Networks customized to storage networking, such as Fiber Channel, were developed largely due to the perception that standard networking protocols are too heavyweight for attaching storage. Conventional wisdom says that IP storage is impractical without special purpose NICs to accelerate the TCP/IP protocol stack. This papers

Paper 41 (036 Reply) at 14; Paper 46 (072 Reply) at 13-14;
Ex. 2300 (IP Storage and the CPU Consumption Myth) at 1.

Dr. Horst's article was focused on networked storage, not other markets

IP Storage and the CPU Consumption Myth

Robert Horst
3ware, Inc.
701 E. Middlefield Rd.
Mountain View, CA 94043

Abstract

This paper addresses a key issue t attaching storage devices directly to t perceived need for hardware acceleratio networking stack. While many implic acceleration is required, the evidence conclusion is not well founded. In th accelerators have had mixed success, economic justification for hardware acc given the low cost of host CPU cycles. many applications is dominated by a transfer rate, and hardware protocol a little effect on the I/O performance in the Application benchmarks were run on subsystem to measure performance and on Email, database, file serving, and bac The results show that good performance without protocol acceleration.

1. Introduction

The growing popularity of gigab prompted increasing interest in usi networks to attach storage devices to Ethernet Storage Area Networks (significant advantages in cost and m compared with Fibre Channel SANs. products are already on the market standardize the protocols is progressing working group of the IETF [1].

Networks customized to storage net Fiber Channel, were developed larg perception that standard networking p heavyweight for attaching storage. Conv says that IP storage is impractical withou NICs to accelerate the TCP/IP protocol st shows that the need for hardware accelera myth. Several different lines of reasoni future of storage networking will rely h devices connected to servers without hardware accelerators.

The growing popularity of gigabit Ethernet has prompted increasing interest in using standard IP networks to attach storage devices to servers. These Ethernet Storage Area Networks (E-SANs), have significant advantages in cost and management ease compared with Fibre Channel SANs. Some IP storage products are already on the market, and work to standardize the protocols is progressing in the IP Storage working group of the IETF [1].

Paper 41 (036 Reply) at 14-15; Paper 46 (072 Reply) at 14;
Ex. 2300 (IP Storage and the CPU Consumption Myth) at 1.

036 Patent: Disputes

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

   a) **<u>The prior art: "said communication processing mechanism containing a second processor" (all claims)</u>**

   b) The prior art discloses "[second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory" (all claims)

   c) The prior art discloses "the TCP state information is updated by said second processor" (all claims)

   d) The prior art discloses a "receive sequencer with directions to classify said packet" on "said communication processing mechanism" (claim 2)

   e) The prior art discloses a "receive sequencer with directions to generate a summary" on "said communication processing mechanism (claim 3)

“Said communication processing mechanism containing a second processor”

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING 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 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538 A 12/1982 Johnson et al. .....

(Continued)

FOREIGN PATENT DOCUMENT

WO WO/98/19412 5/1998

(Continued)

OTHER PUBLICATIONS

Internet pages entitled “Hardware Assisted Protoco
(which Eugene Feinberg is working on), 1 page, pr
1998.

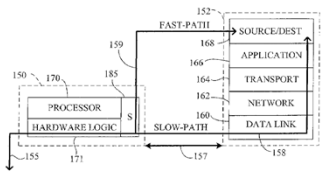
(Continued)

Primary Examiner—Jeffrey Pwu
Assistant Examiner—Jude Jean-Gilles
(74) *Attorney, Agent, or Firm*—Mark Lauer;
Law Group LLP

(57) **ABSTRACT**

A system for protocol processing in a computer
an intelligent network interface card (INIC) or
tion processing device (CPD) associated with
puter. The INIC provides a fast-path that avo
processing for most large multi-packet mess
accelerating data communication. The INIC al
host for those message packets that are chosen
ing by host software layers. A communication
for a message is defined that allows DMA cont
INIC to move data, free of headers, directly
destination or source in the host. The context is
INIC as a communication control block (CCB
passed back to the host for message processing
The INIC contains specialized hardware circ
much faster at their specific tasks than a gen
CPU. A preferred embodiment includes a trio
processors with separate processors devoted
receive and management processing, with full
munication for four fast Ethernet nodes.

22 Claims, 89 Drawing Sheets

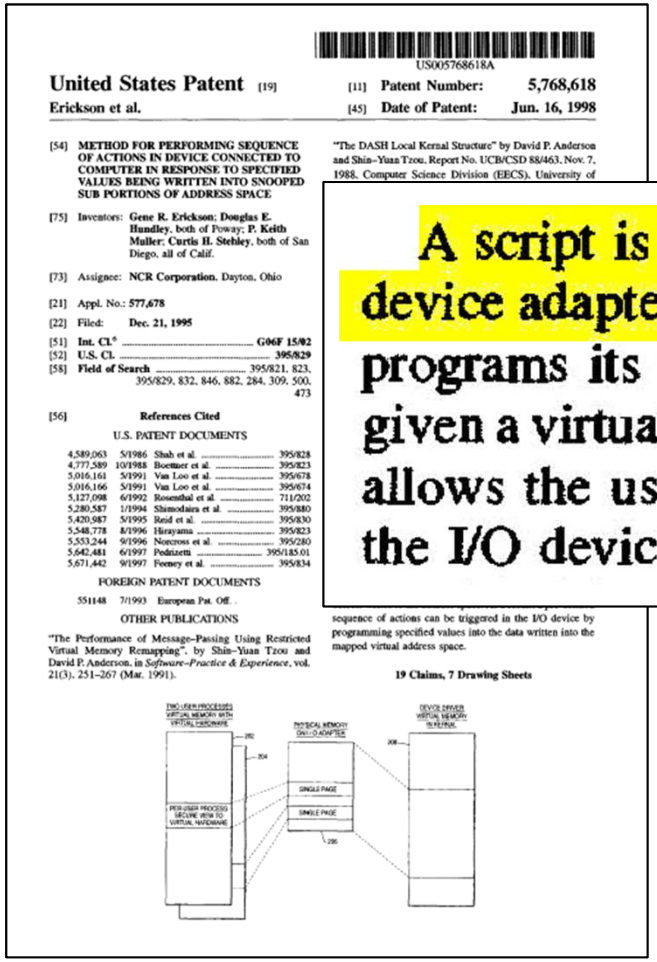


1. A device for use with a first apparatus that is connectable to a second apparatus, the first apparatus containing a memory and a first processor operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:

a communication processing mechanism connected to the first processor, said communication processing mechanism containing a second processor running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory and the TCP state information is updated by said second processor.

Ex. 1001 (036 Patent), Claims 1-7.

Erickson discloses a second processor



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

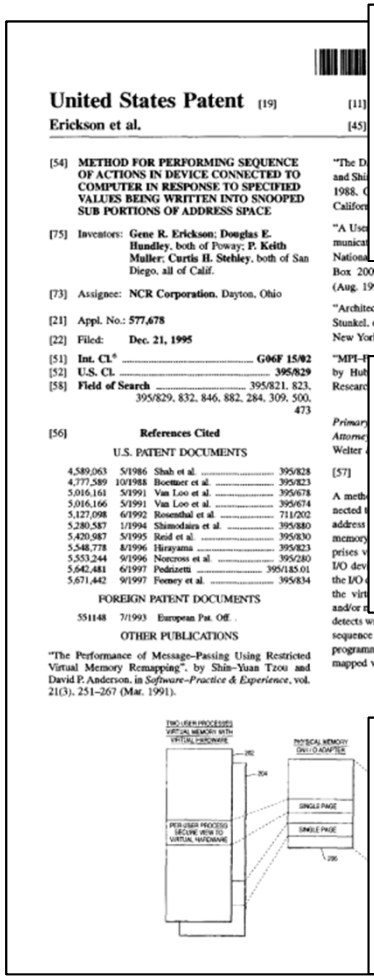
sequence of actions can be triggered in the I/O device by programming specified values into the data written into the mapped virtual address space.

19 Claims, 7 Drawing Sheets

A script is prepared by the operating system for the I/O device adapter to execute each time the specific user process programs its specific virtual hardware. The user process is given a virtual address in the user process' address space that allows the user process very specific access capabilities to the I/O device adapter.

Paper 2 (036 Petition) at 65;
Paper 41 (036 Reply) at 15;
Ex. 1005 (Erickson) at 4:18-23.

Erickson shows second processor on adapter executes scripts



United States Patent [19]
Erickson et al.

[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. Stebley, 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 D... and Shi... 1988, C... Califor...

"A Use... municat... Nationa... 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... by Hub... Researc...

Primary... Attorne... Welter ...

[57] A meth... nected t... address... memory... prises v... I/O dev... the I/O... the virt... and/or r... 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

memory on the I/O device adapter. If the I/O device adapter detects access to the physical memory page, a predefined script is then executed by the I/O device adapter in order to direct the data as appropriate.

Paper 41 (036 Reply) at 16;
Ex. 1005 (Erickson) at 5:37-40.

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.

Paper 41 (036 Reply) at 16;
Ex. 1005 (Erickson) at 7:41-44.

exception for the system bus controller. The bus controller then transfers the data to the I/O device adapter and initiates the registers of the I/O device adapter to execute a predetermined script to process the data.

Paper 41 (036 Reply) at 16;
Ex. 1005 (Erickson) at 8:54-57.

036 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 “said communication processing mechanism containing a second processor” (all claims)

   b) <u>The prior art discloses “[second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory” (all claims)</u>

   c) The prior art discloses “the TCP state information is updated by said second processor” (all claims)

   d) The prior art discloses a “receive sequencer with directions to classify said packet” on “said communication processing mechanism” (claim 2)

   e) The prior art discloses a “receive sequencer with directions to generate a summary” on “said communication processing mechanism (claim 3)

“[Second processor] running instructions to process a message packet...”

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING 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 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538 A 12/1982 Johnson et al. .....

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO 98/19412 5/1998

(Continued)

OTHER PUBLICATIONS

Internet pages entitled “Hardware Assisted Protocol
(which Eugene Feinberg is working on), 1 page, print
1998.

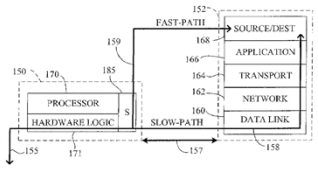
(Continued)

Primary Examiner—Jeffrey Pwu
Assistant Examiner—Jude Jean-Gilles
(74) *Attorney, Agent, or Firm*—Mark Lauer; Si
Law Group LLP

(57) **ABSTRACT**

A system for protocol processing in a computer n
an intelligent network interface card (INIC) or c
tion processing device (CPD) associated with a
puter. The INIC provides a fast-path that avoi
processing for most large multi-packet messag
accelerating data communication. The INIC also
host for those message packets that are chosen f
ing by host software layers. A communication co
for a message is defined that allows DMA contro
INIC to move data, free of headers, directly to
destination or source in the host. The context is s
INIC as a communication control block (CCB)
passed back to the host for message processing b
The INIC contains specialized hardware circui
much faster at their specific tasks than a gener
CPU. A preferred embodiment includes a trio o
processors with separate processors devoted t
receive and management processing, with full d
munication for four fast Ethernet nodes.

22 Claims, 89 Drawing Sheets



1. A device for use with a first apparatus that is connectable to a second apparatus, the first apparatus containing a memory and a first processor operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:

a communication processing mechanism connected to the first processor, said communication processing mechanism containing a second processor running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory and the TCP state information is updated by said second processor.

Ex. 1001 (036 Patent), Claim 1.

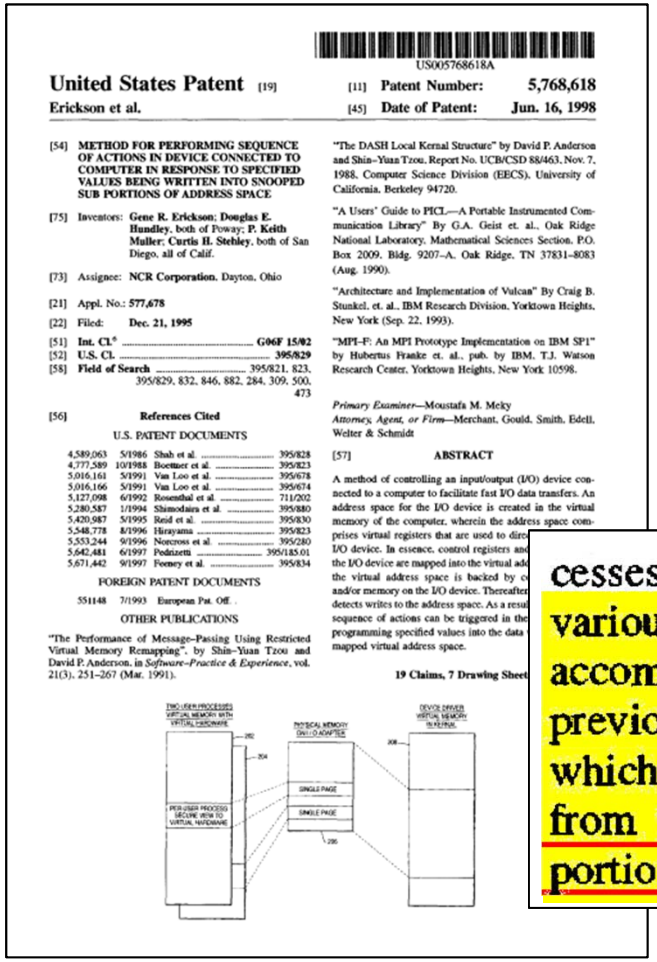
PO does not address prior art combination petitioner relies on

U.S. Pat. No. 5,768,618 ("Erickson") in view of Tanenbaum96
[1.3] [second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory and

Accordingly, Erickson in view of Tanenbaum96 disclose *running instructions* (specified by the scripts) *to process a message packet such that the context* (including the script, registers 508 and 504, endpoint table 514, endpoint protocol data 518, pre-negotiated information, and pointer to main memory) *is employed* (to identify the relevant protocol script to run, and further to identify where to write the received data) *to transfer data contained in said packet to the first apparatus memory* (memory of host computer)

Paper 2 (036 Petition) at 66-68;
Paper 41 (036 Reply) at 17;
Ex. 1003.109 (036 Horst Decl.).

Erickson's adapter stores protocol information for moving data to the host



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).

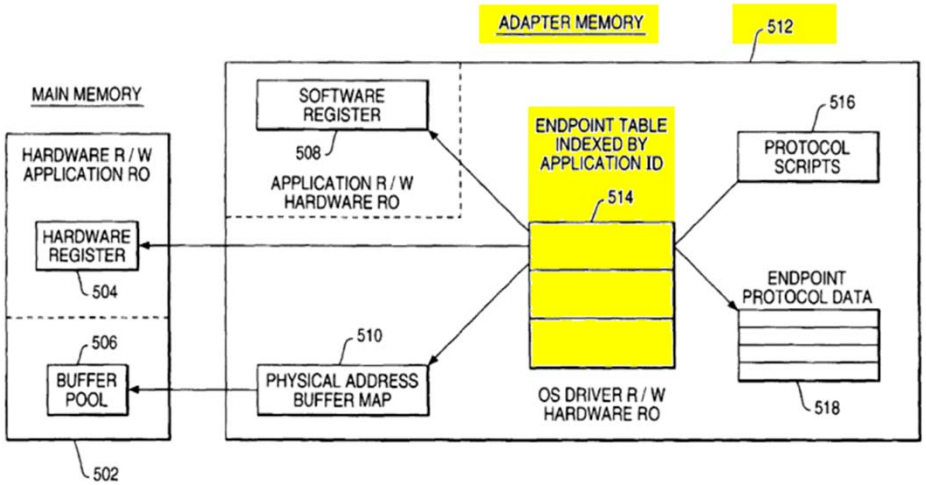
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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

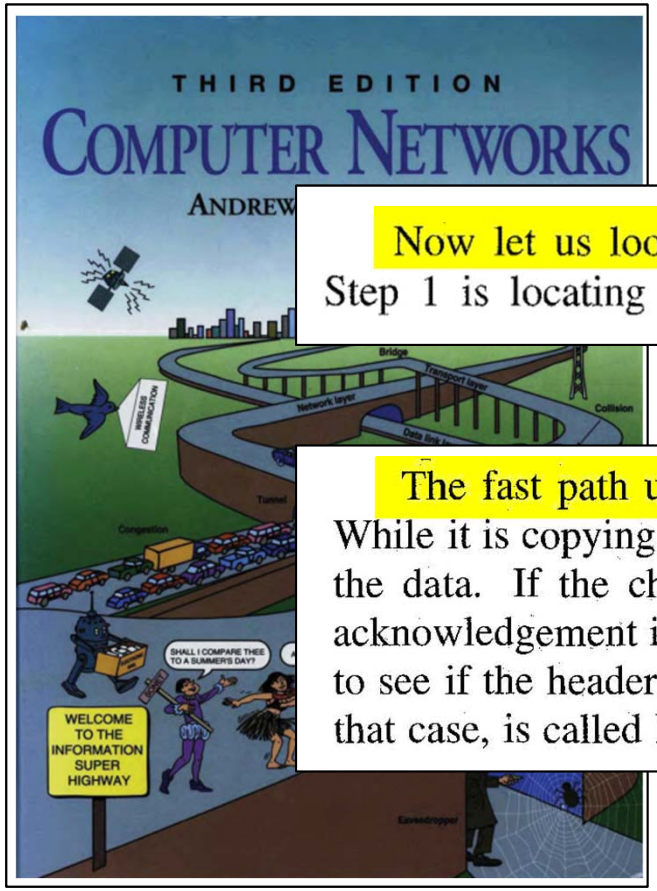
19 Claims, 7 Drawing Sheet



cesses. Each entry within the endpoint table **514** points to various protocol data **518** in the memory **512** in order to accommodate multiple communication protocols, as well as previously defined protocol scripts **516** in the memory **512**, which indicate how data or information is to be transferred from the memory **512** of the I/O device adapter to the portions of main memory **502** associated with a user process.

Paper 2 (036 Petition) at 67;
Ex. 1005 (Erickson) at 5:61-67.

It would be obvious to use Tanenbaum96's fast-path connection records with Erickson



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 2 (036 Petition) at 60-61;
Paper 41 (036 Reply) at 17;
Ex. 1003.098-.099 (036 Horst Decl.) at A-12 –A-13;
Ex. 1006.584-.585 (Tanenbaum96).

036 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 “said communication processing mechanism containing a second processor” (all claims)

   b) The prior art discloses “[second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory” (all claims)

   c) <u>The prior art discloses “the TCP state information is updated by said second processor” (all claims)</u>

   d) The prior art discloses a “receive sequencer with directions to classify said packet” on “said communication processing mechanism” (claim 2)

   e) The prior art discloses a “receive sequencer with directions to generate a summary” on “said communication processing mechanism (claim 3)

“The TCP state information is updated by said second processor”

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING 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 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed on Mar. 6, 2002, now Pat. No. 6,591,302, which is a

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,366,538 A 12/1982 Johnson et al. .....

(Continued)

FOREIGN PATENT DOCUMENTS

WO WO/98/19412 5/1998

(Continued)

OTHER PUBLICATIONS

Internet pages entitled “Hardware Assisted Protoco
(which Eugene Feinberg is working on), 1 page, pri
1998.

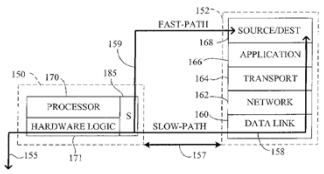
(Continued)

Primary Examiner—Jeffrey Pwu
Assistant Examiner—Jude Jean-Gilles
(74) *Attorney, Agent, or Firm*—Mark Lauer;
Law Group LLP

(57) **ABSTRACT**

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

22 Claims, 89 Drawing Sheets

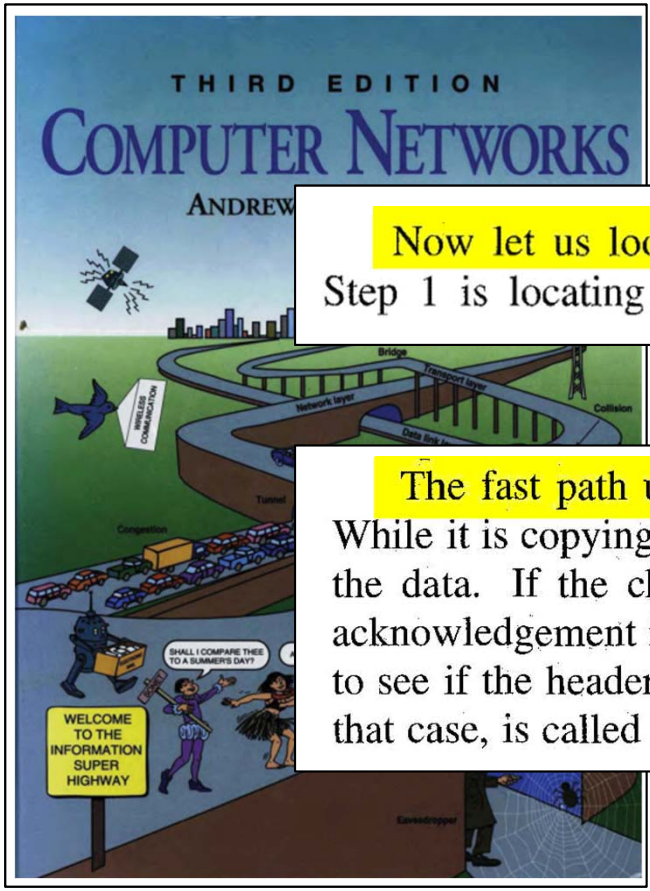


1. A device for use with a first apparatus that is connectable to a second apparatus, the first apparatus containing a memory and a first processor operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:

a communication processing mechanism connected to the first processor, said communication processing mechanism containing a second processor running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory and the TCP state information is updated by said second processor.

Ex. 1001 (036 Patent), Claim 1.

It would be obvious to use Tanenbaum96's fast-path connection records with Erickson

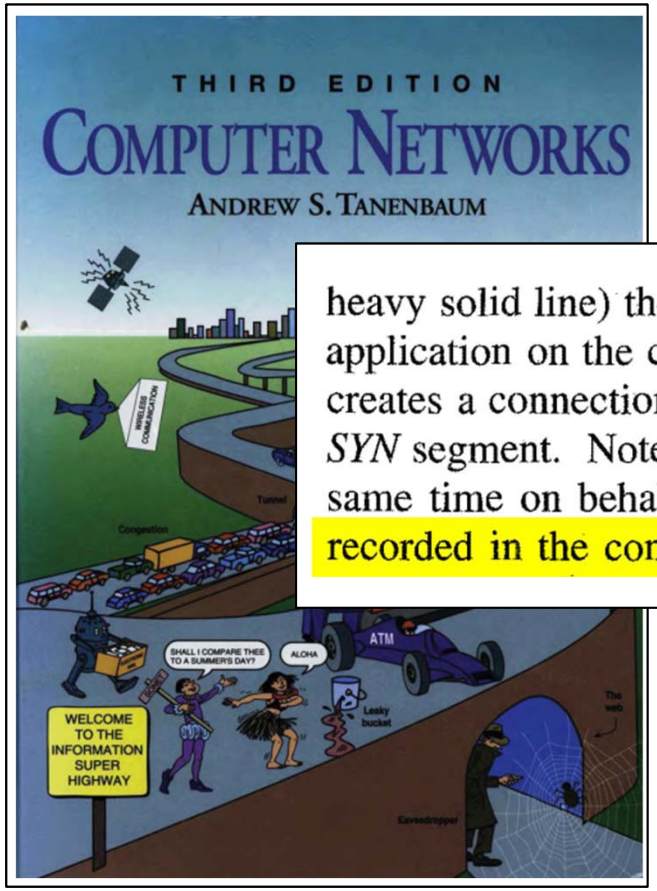


Now let us look at fast path processing on the receiving side of Fig. 6-49. Step 1 is locating the connection record for the incoming TPDU. For ATM,

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 2 (036 Petition) at 68-69;
Ex. 1003.110 (036 Horst Decl.);
Ex. 1006.584-.585 (Tanenbaum96).

Tanenbaum96: TCP state information is stored in a connection record



heavy solid line) then later the path of a server (the heavy dashed line). When an application on the client machine issues a CONNECT request, the local TCP entity creates a connection record, marks it as being in the *SYN SENT* state, and sends a *SYN* segment. Note that many connections may be open (or being opened) at the same time on behalf of multiple applications, so the state is per connection and recorded in the connection record. When the *SYN+ACK* arrives, TCP sends the

Paper 2 (036 Petition) at 68-69;
Ex. 1003.110-.111 (036 Horst Decl.)
Ex. 1006.549 (Tanenbaum96).

036 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 “said communication processing mechanism containing a second processor” (all claims)

   b) The prior art discloses “[second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory” (all claims)

   c) The prior art discloses “the TCP state information is updated by said second processor” (all claims)

   d) <u>The prior art discloses a “receive sequencer with directions to classify said packet” on “said communication processing mechanism” (claim 2)</u>

   e) The prior art discloses a “receive sequencer with directions to generate a summary” on “said communication processing mechanism (claim 3)

“Receive sequencer with directions to classify said packet”

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING 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. Higgs**, Saratoga, CA (US); **Clive M. Philbr**, 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 patent is extended or adjusted under U.S.C. 154(b) by 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed Mar. 6, 2002, now Pat. No. 6,591,302, which

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52

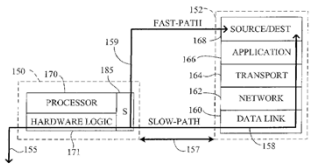
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

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.

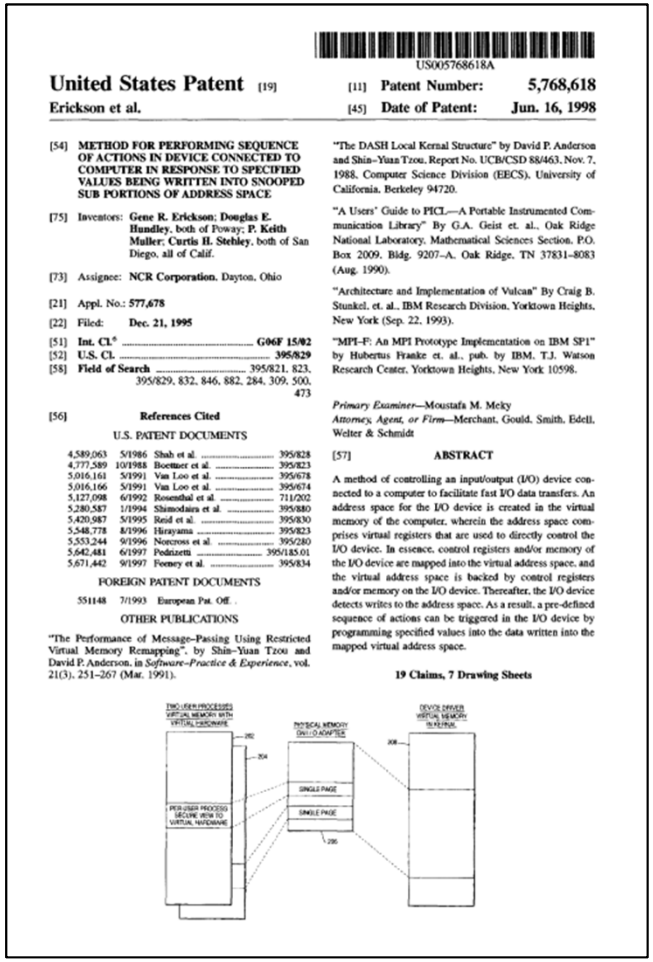
22 Claims, 89 Drawing Sheets



2. The device of claim **1**, wherein said communication processing mechanism includes a receive sequencer with directions to classify said packet, wherein said packet contains control information corresponding to the stack of protocol layers.

Ex. 1001 (036 Patent), Claim 2.

Erickson's adapter classifies received packets by application and protocol



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	Pedzierzi	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).

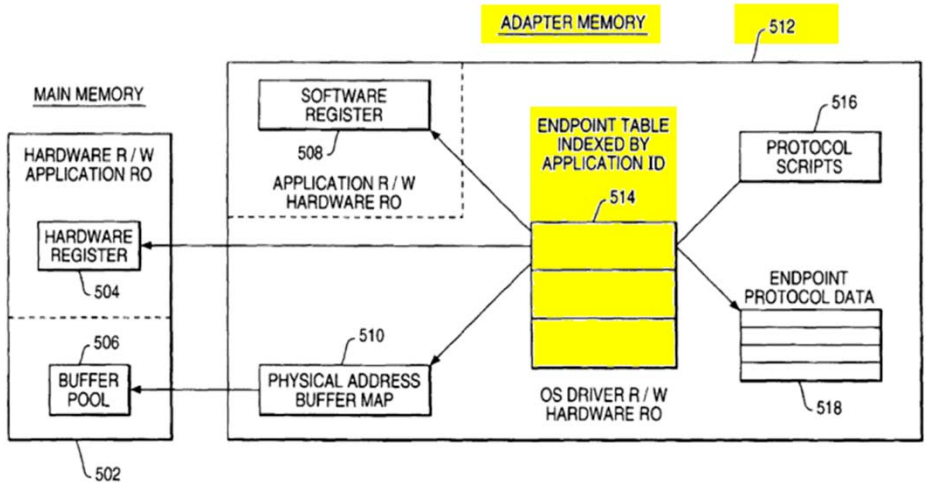
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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.

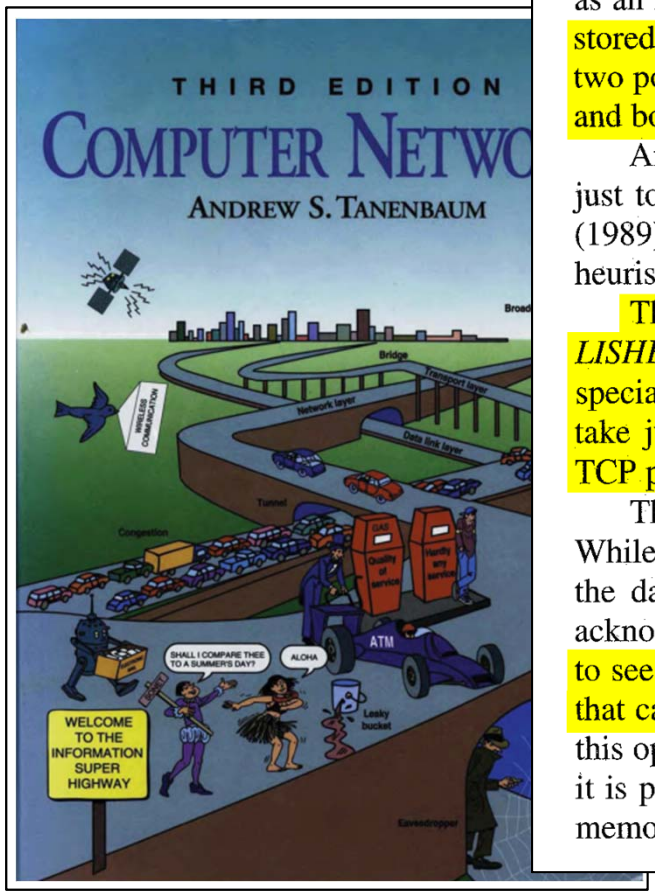
19 Claims, 7 Drawing Sheets



Protocol scripts typically serve two functions. The first function is to describe the protocol the software application is using. This includes but is not limited to how to locate an application endpoint, and how to fill in a protocol header template from the application specific data buffer. The second function is to define a particular set of instructions to 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 2 (036 Petition) at 70; Paper 41 (036 Reply) at 18-19;
Ex. 1003.112 (036 Horst Decl.); Ex. 1005 (Erickson) at 5:41-51.

Tanenbaum96: Receive sequencer receives and classifies packets



as an index to find the connection record. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses and both ports must be compared to verify that the correct record has been found.

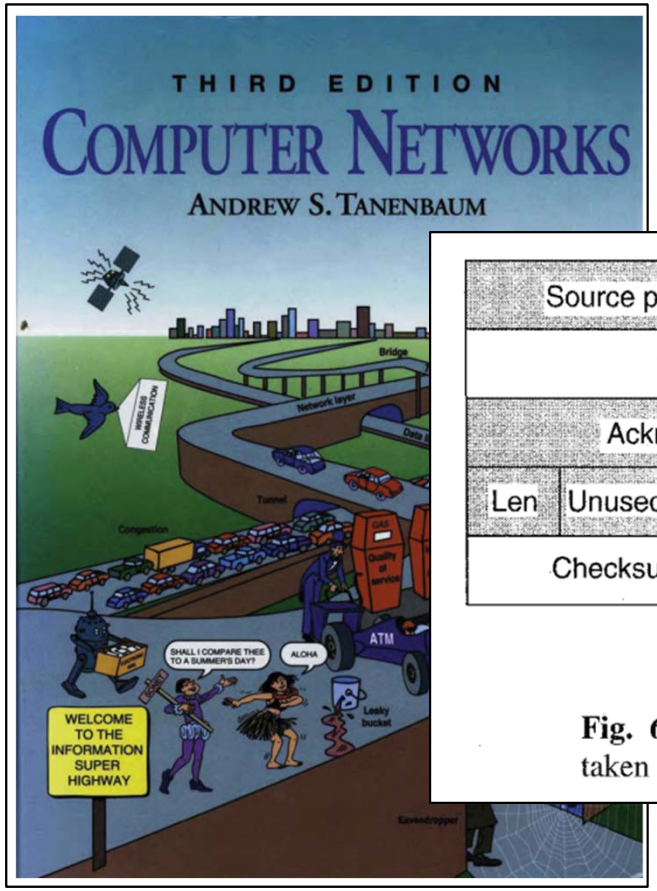
An optimization that often speeds up connection record lookup even more is just to maintain a pointer to the last one used and try that one first. Clark et al. (1989) tried this and observed a hit rate exceeding 90 percent. Other lookup heuristics are described in (McKenney and Dove, 1992).

The TPDU is then checked to see if it is a normal one: the state is *ESTABLISHED*, neither side is trying to close the connection, the TPDU is a full one, no special flags are set, and the sequence number is the one expected. These tests take just a handful of instructions. If all conditions are met, a special fast path TCP procedure is called.

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 this optimization and all the other ones discussed in this chapter are used together, it is possible to get TCP to run at 90 percent of the speed of a local memory-to-memory copy, assuming the network itself is fast enough.

Paper 2 (036 Petition) at 70-71; Paper 41 (036 Reply) at 18-19;
Ex. 1003.112-.114 (036 Horst Decl.); Ex. 1006.584-.585 (Tanenbaum96).

Tanenbaum96: TCP packet contains control information



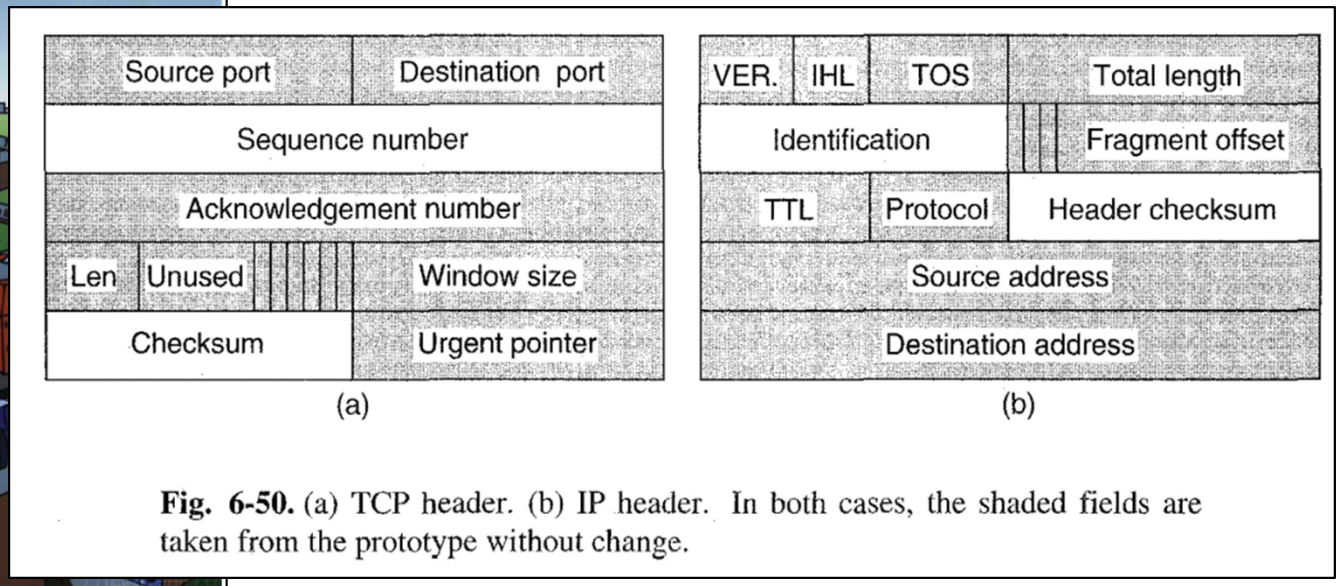


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 2 (036 Petition) at 72; Paper 41 (036 Reply) at 18-19;
Ex. 1003.114-.115 (036 Horst Decl.); Ex. 1006.584 (Tanenbaum96).

036 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 “said communication processing mechanism containing a second processor” (all claims)

   b) The prior art discloses “[second processor] running instructions to process a message packet such that the context is employed to transfer data contained in said packet to the first apparatus memory” (all claims)

   c) The prior art discloses “the TCP state information is updated by said second processor” (all claims)

   d) The prior art discloses a “receive sequencer with directions to classify said packet” on “said communication processing mechanism” (claim 2)

   e) <u>The prior art discloses a “receive sequencer with directions to generate a summary” on “said communication processing mechanism (claim 3)</u>

“Receive sequencer with directions to generate a summary”

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

(10) **Patent No.: US 7,237,036 B2**
(45) **Date of Patent: Jun. 26, 2007**

(54) **FAST-PATH APPARATUS FOR RECEIVING DATA CORRESPONDING 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 thi patent is extended or adjusted under 3 U.S.C. 154(b) by 672 days.

(21) Appl. No.: **10/260,112**

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

(65) **Prior Publication Data**

US 2004/0073703 A1 Apr. 15, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/092,967, filed o Mar. 6, 2002, now Pat. No. 6,591,302, which is

(Continued)

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

(51) **Int. Cl.**
G06F 13/38 (2006.01)
G06F 15/17 (2006.01)

(52) **U.S. Cl.** **709/245**; 709/236; 709/230; 370/474; 370/396; 370/469

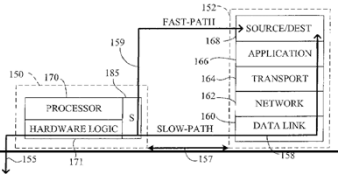
(58) **Field of Classification Search** 709/245, 709/236, 230, 202; 370/474, 230, 396, 469; 707/2–4, 10; 712/19, 52

See application file for complete search history.

(56) **References Cited**

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.

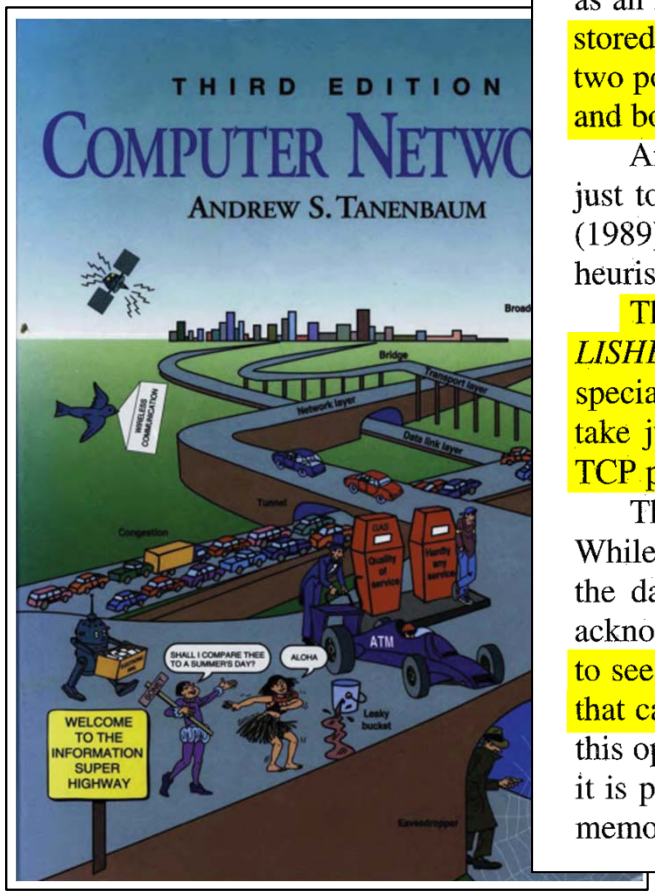
22 Claims, 89 Drawing Sheets



3. The device of claim **1**, wherein said communication processing mechanism includes a receive sequencer with directions to generate a summary of a second message packet received from the network, said second packet containing control information corresponding to the stack of protocol layers, and said instructions including an instruction to compare said summary with said context.

Ex. 1001 (036 Patent), Claim 3.

Tanenbaum96: Summary (IP addresses and ports) compared against context



as an index to find the connection record. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses and both ports must be compared to verify that the correct record has been found.

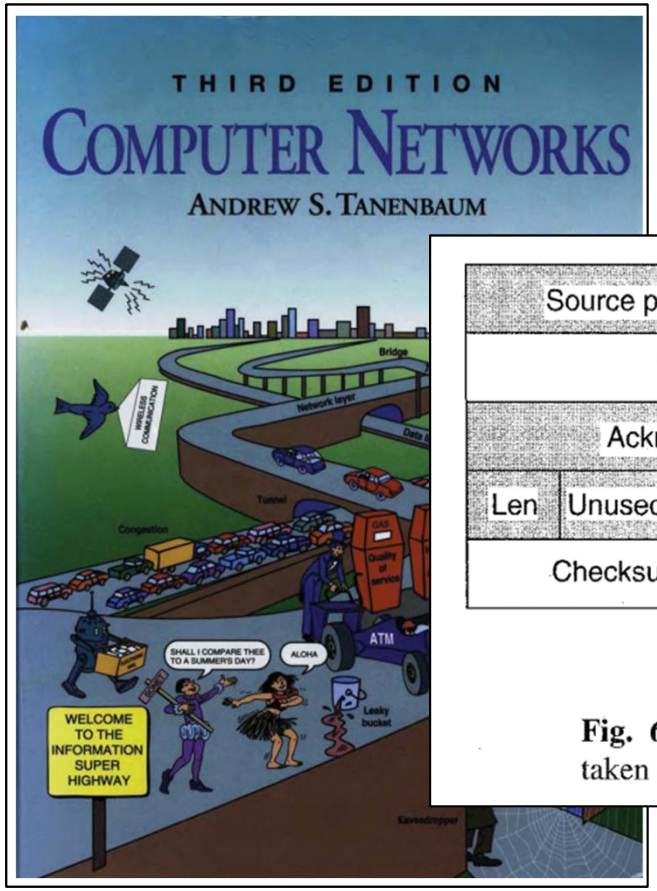
An optimization that often speeds up connection record lookup even more is just to maintain a pointer to the last one used and try that one first. Clark et al. (1989) tried this and observed a hit rate exceeding 90 percent. Other lookup heuristics are described in (McKenney and Dove, 1992).

The TPDU is then checked to see if it is a normal one: the state is *ESTABLISHED*, neither side is trying to close the connection, the TPDU is a full one, no special flags are set, and the sequence number is the one expected. These tests take just a handful of instructions. If all conditions are met, a special fast path TCP procedure is called.

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 this optimization and all the other ones discussed in this chapter are used together, it is possible to get TCP to run at 90 percent of the speed of a local memory-to-memory copy, assuming the network itself is fast enough.

Paper 2 (036 Petition) at 73-74; Paper 41 (036 Reply) at 18-19;
Ex. 1003.117 (036 Horst Decl.); Ex. 1006.584-.585 (Tanenbaum96).

Tanenbaum96: TCP packet contains control information



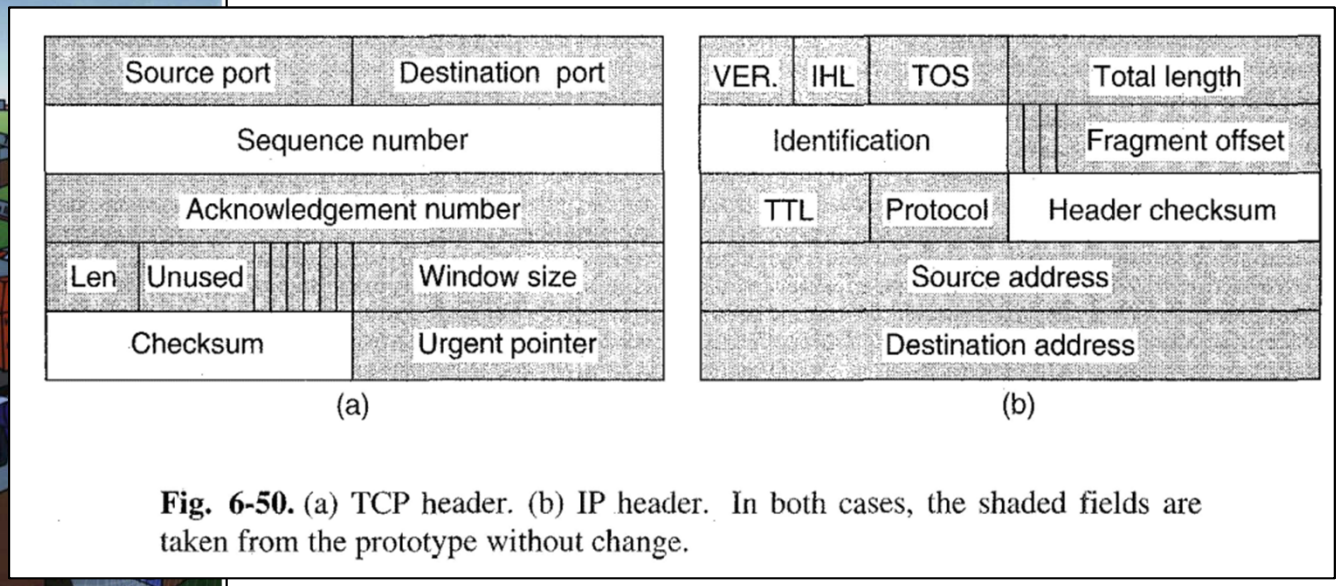
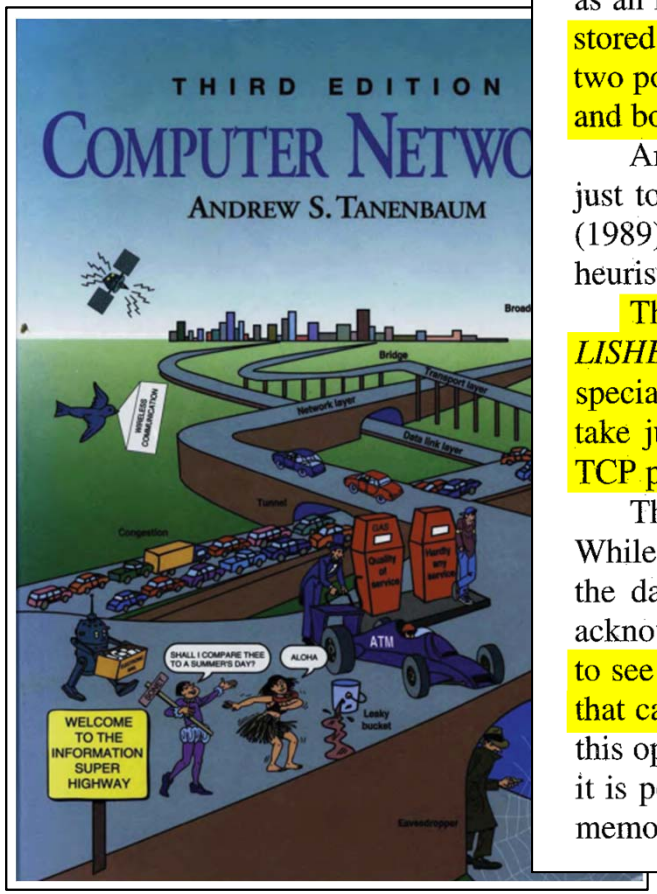


Fig. 6-50. (a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Paper 2 (036 Petition) at 74; Paper 41 (036 Reply) at 18-19;
Ex. 1003.118 (036 Horst Decl.); Ex. 1006.584 (Tanenbaum96).

Tanenbaum96: Comparison of summary to context verifies packet is candidate for fast-path



as an index to find the connection record. For TCP, the connection record can be stored in a hash table for which some simple function of the two IP addresses and two ports is the key. Once the connection record has been located, both addresses and both ports must be compared to verify that the correct record has been found.

An optimization that often speeds up connection record lookup even more is just to maintain a pointer to the last one used and try that one first. Clark et al. (1989) tried this and observed a hit rate exceeding 90 percent. Other lookup heuristics are described in (McKenney and Dove, 1992).

The TPDU is then checked to see if it is a normal one: the state is *ESTABLISHED*, neither side is trying to close the connection, the TPDU is a full one, no special flags are set, and the sequence number is the one expected. These tests take just a handful of instructions. If all conditions are met, a special fast path TCP procedure is called.

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 this optimization and all the other ones discussed in this chapter are used together, it is possible to get TCP to run at 90 percent of the speed of a local memory-to-memory copy, assuming the network itself is fast enough.

Paper 2 (036 Petition) at 73-75; Paper 41 (036 Reply) at 18-19;
Ex. 1003.117-119 (036 Horst Decl.); Ex. 1006.584-.585 (Tanenbaum96).

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

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

b) Patent Owner does not show adequate written description support

c) Substitute claims are indefinite

d) Substitute claims are obvious

Substitute claim 23 adds an alternative to a recited step in claim 1

Original Claim 1	Substitute Claim 23
[1.P.1]. A device for use with a first apparatus that is connectable to a second apparatus,	[23.P.1] A device for use with a first apparatus that is connectable to a second apparatus,
[1.P.2]. the first apparatus containing a memory and a first processor	[23.P.2] the first apparatus containing a memory and a first processor
[1.P.3] operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:	[23.P.3] operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:
[1.1] a communication processing mechanism connected to the first processor,	[23.1] a communication processing mechanism connected to the first processor,
[1.2] said communication processing mechanism containing a second processor	[23.2] said communication processing mechanism containing a second processor
[1.3] running instructions	[23.3] running instructions
to process a message packet	
	on the second processor, wherein the second processor determining whether an incoming message packet should be processed by the second processor,
	[23.4] if the incoming message packet should be processed by the second processor, processing the incoming message packet, without involving the stack of processing protocol processing layers,
such that the context is employed to transfer data contained in said packet to the first apparatus memory and	such that the context is employed to transfer data contained in said packet to the first apparatus memory and
[1.4] the TCP state information is updated by said second processor.	[23.5] the TCP state information is updated by said second processor,
	[23.6] if the incoming message packet should not be processed by the second processor, passing the incoming message packet to the first processor for further processing.

Requires "fast-path"

Requires either "fast path" **or** "slow path"

Paper 36 (036 Opp. to Motion to Amend) at 6.

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

a) Patent Owner has improperly expanded the scope of the claims

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

c) Substitute claims 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 50 (036 Sur-Reply for Motion to Amend) at 3;
Nov. 21, 2017 USPTO Memo

PO identifies same 10 pages and 12 figures from original disclosure

Claims	Exemplary Support in the '112 Application
Proposed Claim 23	
~~1~~23. A device for use with a first apparatus that is connectable to a second apparatus, the first apparatus containing a memory and a first processor operating a stack of protocol processing layers that create a context for communication, the context including a media access control (MAC) layer address, an Internet Protocol (IP) address and Transmission Control Protocol (TCP) state information, the device comprising:	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
a communication processing mechanism connected to the first processor,	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
said communication processing mechanism containing a second processor	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
running instructions on the second processor, wherein the second processor determining whether an incoming message packet should be processed by the second processor,	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.
if the incoming message packet should be processed by the second processor, ~~to~~ processing the ~~a~~ incoming message packet, without involving the stack of processing protocol processing layers, such that the context is employed to transfer data contained in said packet to the first apparatus memory and the TCP state information is updated by said second processor,	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1..
if the incoming message packet should not be processed by the second processor, passing the incoming message packet to the first processor for further processing.	*See*, *e.g.*, Ex. 2020 at Abstract, Figs. 1-3, 4A-4D, and 5-12, Pages 7-8, 10-17, Cl. 1.

Same written description support as 072 Patent

Paper 21 (036 Motion to Amend) Appendix A at i-ii.

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**
>
> 65. It is my opinion that the proposed amendments are supported by the written description, along with the Application (No. 10/260,112) and Provisional Application (No. 60/061,809).

Paper 42 (036 Reply ISO Motion to Amend) at 6;
Ex. 2305 (Almeroth Decl. ISO Reply) at 22.

Written description support provided by PO is insufficient

- Patent Owner cites to written description support not included in its original motion (e.g., pages 18-21 of Ex. 2020)

- Patent Owner has not identified any written description support for:
  - "running instructions on the second processor, wherein the second processor determining..."
  - "such that the context [including a MAC layer address] is employed to transfer data"

Paper 50 (036 Sur-Reply for Motion to Amend) at 5-6.

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

a) Patent Owner has improperly expanded the scope of the claims

b) Patent Owner does not show adequate written description support

c) <u>Substitute claims are indefinite</u>

d) Substitute claims are obvious

Substitute claim 23 requires processing to determine whether to continue processing

23.3) **running instructions on the second processor,** wherein the second processor determining whether an incoming message packet **should be processed by the second processor**,

Paper 36 (036 Opp. to Motion to Amend) at 10-11.

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

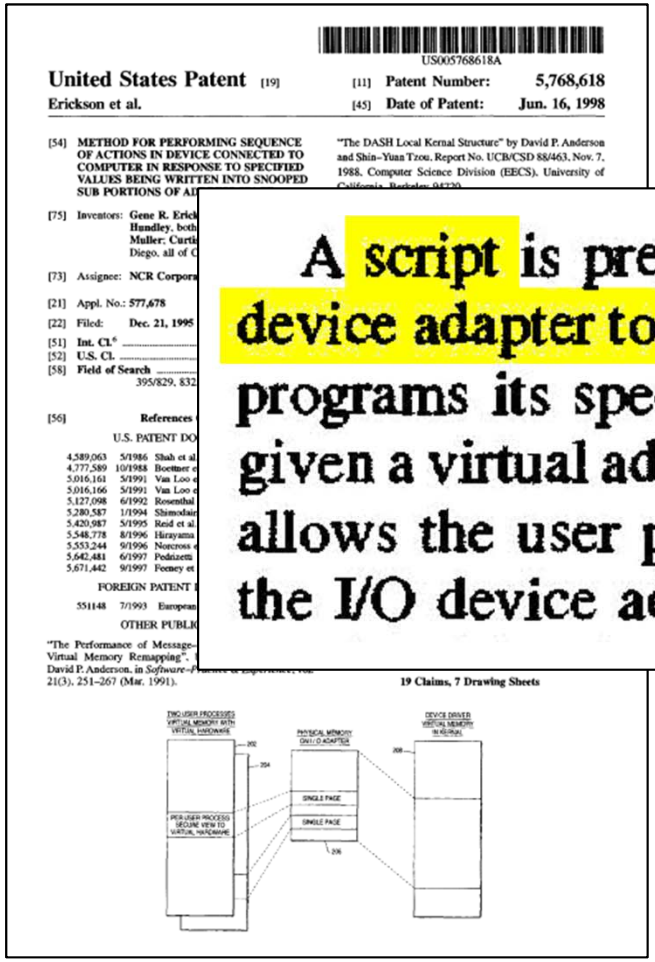
c) Substitute claims are obvious

i. **<u>Prior art discloses "the second processor determining whether an incoming message packet should be processed by second processor" (limitation 23.3)</u>**

ii. Prior art discloses "processing the incoming message packet [by the second processor]" (limitation 23.4)

iii. Prior art discloses "passing the incoming message packet to the first processor for further processing" (limitation 23.5)

Erickson discloses a second processor



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 AL

[75] Inventors: Gene R. Erick Hundley, both Muller; Curti Diego, all of C

[73] Assignee: NCR Corpora

[21] Appl. No.: 577,678

[22] Filed: Dec. 21, 1995

[51] Int. Cl.[6]

[52] U.S. Cl.

[58] Field of Search 395/829, 832

[56] References

U.S. PATENT DO

4,589,063	5/1986	Shah et al
4,777,589	10/1988	Boettner e
5,016,161	5/1991	Van Loo e
5,016,166	5/1991	Van Loo e
5,127,098	6/1992	Rosenthal
5,280,587	1/1994	Shimodaic
5,420,987	5/1995	Reid et al
5,548,778	8/1996	Hirayama
5,553,244	9/1996	Norcross e
5,642,481	6/1997	Pedrizetti
5,671,442	9/1997	Feeney et

FOREIGN PATENT

551148 7/1993 European

OTHER PUBLI

"The Performance of Message-Virtual Memory Remapping", David P. Anderson, in *Software-P* 21(3), 251–267 (Mar. 1991).

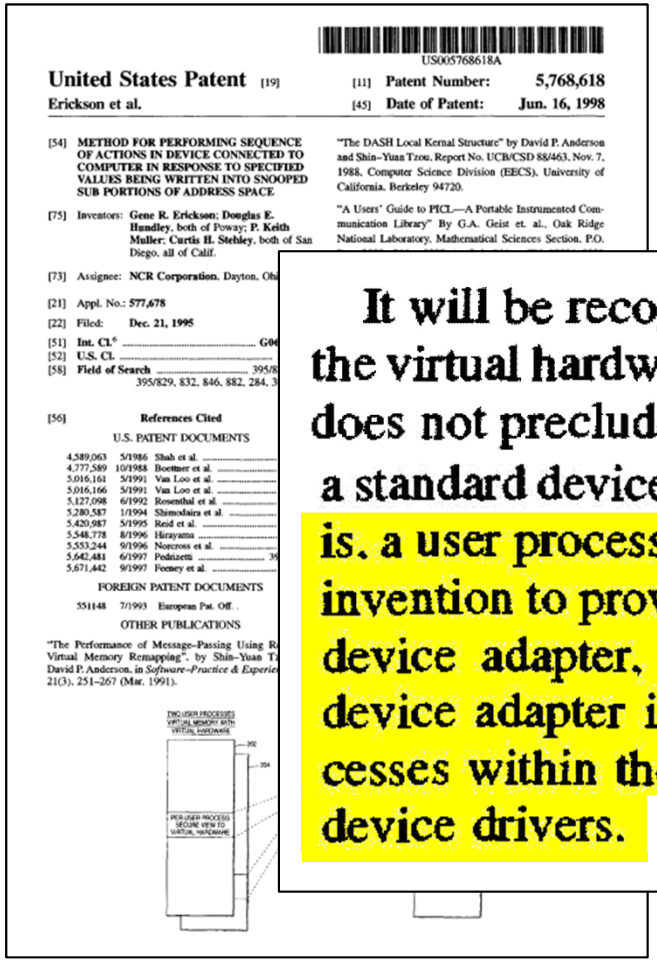
"The DASH Local Kernal Structure" by David P. Anderson and Shin-Yuan Tzou, Report No. UCB/CSD 88/463, Nov. 7, 1988, Computer Science Division (EECS), University of California, Berkeley 94720.

19 Claims, 7 Drawing Sheets

A script is prepared by the operating system for the I/O device adapter to execute each time the specific user process programs its specific virtual hardware. The user process is given a virtual address in the user process' address space that allows the user process very specific access capabilities to the I/O device adapter.

Paper 36 (036 Opp. to Motion to Amend) at 13;
Ex. 1005 (Erickson) at 4:18-23.

Slow and fast applications can be used simultaneously



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, Oh

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

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

[51] **Int. Cl.**[6] G0

[52] **U.S. Cl.** .....

[58] **Field of Search** 395/8
395/829, 832, 846, 882, 284, 3

[56] **References Cited**

U.S. PATENT DOCUMENTS

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

FOREIGN PATENT DOCUMENTS

551148 7/1993 European Pat. Off. .

OTHER PUBLICATIONS

"The Performance of Message-Passing Using R
Virtual Memory Remapping", by Shin-Yuan T
David P. Anderson, in *Software-Practice & Experie*
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.

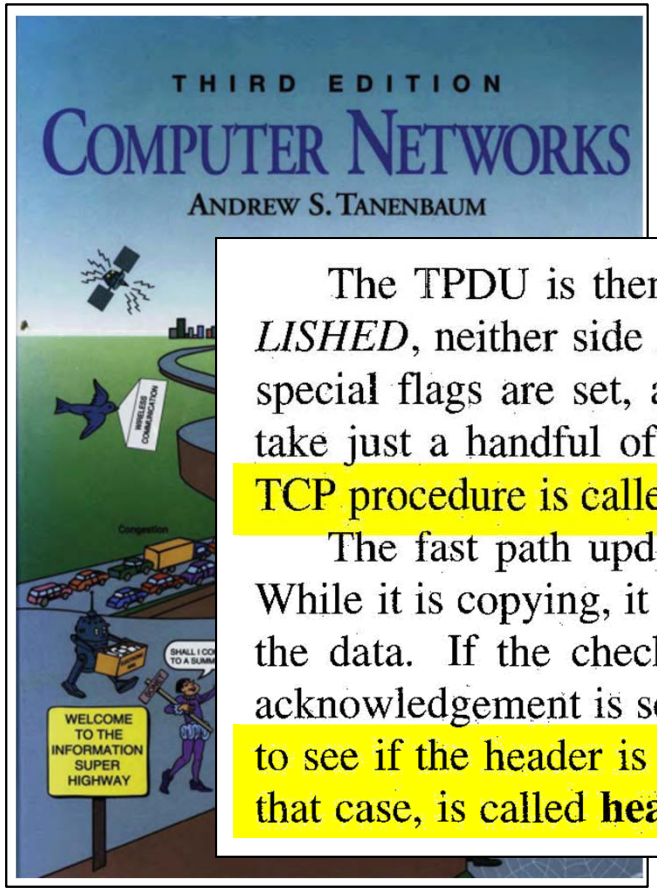
"A Users' Guide to PICL—A Portable Instrumented Communication Library" By G.A. Geist et. al., Oak Ridge National Laboratory, Mathematical Sciences Section, P.O.

It will be recognized by those skilled in the art that using the virtual hardware implementation of the present invention does not preclude a user process from simultaneously using a standard device driver on the same I/O device adapter. That is, a user process can use the virtual hardware of the present invention to provide a direct streamlined path to a given I/O device adapter, while standard slower access to the I/O device adapter is concurrently provided to other user processes within the same system by normal streams or other device drivers.

"Determination" must be made between two applications

Paper 36 (036 Opp. to Motion to Amend) at 14;
Ex. 1005 (Erickson) at 8:65-9:7.

POSA would place header prediction on Erickson's adapter



The TPDU is then checked to see if it is a normal one: the state is *ESTABLISHED*, neither side is trying to close the connection, the TPDU is a full one, no special flags are set, and the sequence number is the one expected. These tests take just a handful of instructions. If all conditions are met, a special fast path TCP procedure is called.

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 36 (036 Opp. to Motion to Amend) at 14-15;
Ex. 1006.585 (Tanenbaum96).

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

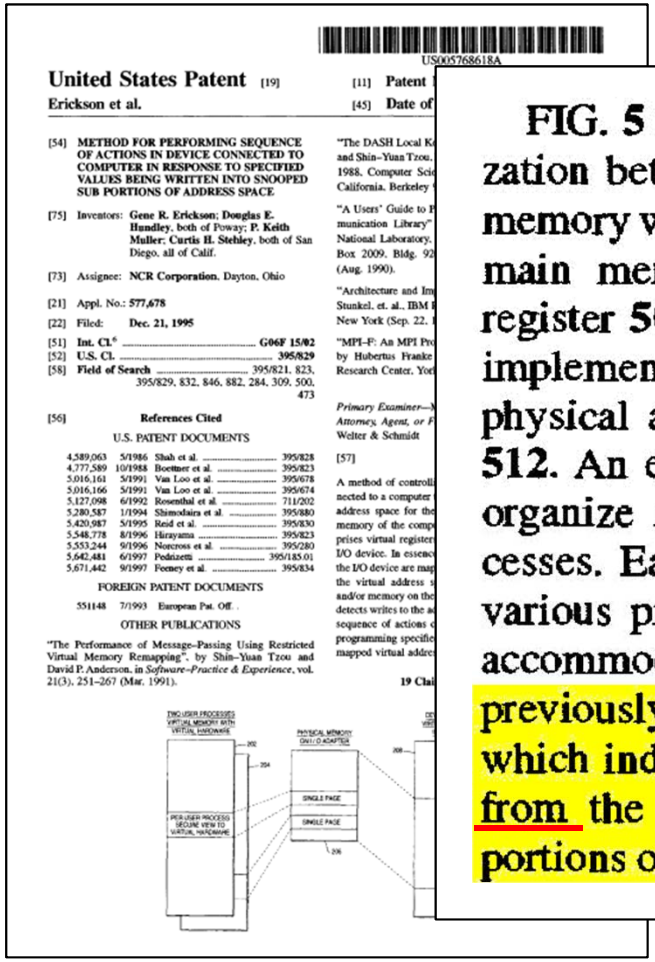
c) Substitute claims are obvious

i. Prior art discloses "the second processor determining whether an incoming message packet should be processed by second processor" (limitation 23.3)

ii. <u>Prior art discloses "processing the incoming message packet [by the second processor]" (limitation 23.4)</u>

iii. Prior art discloses "passing the incoming message packet to the first processor for further processing" (limitation 23.5)

Second processor in Erickson copies data from I/O adapter to host



US005768618A

United States Patent [19]

Erickson et al.

[11] Patent

[45] Date of

[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 K
and Shin-Yuan Tzou.
1988, Computer Sci
California, Berkeley

"A Users' Guide to P
munication Library"
National Laboratory,
Box 2009, Bldg. 92
(Aug. 1990).

"Architecture and Im
Stunkel, et. al., IBM
New York (Sep. 22,

"MPI-F: An MPI Pro
by Hubertus Franke
Research Center, Yor

Primary Examiner—
Attorney, Agent, or F
Welter & Schmidt

[57]

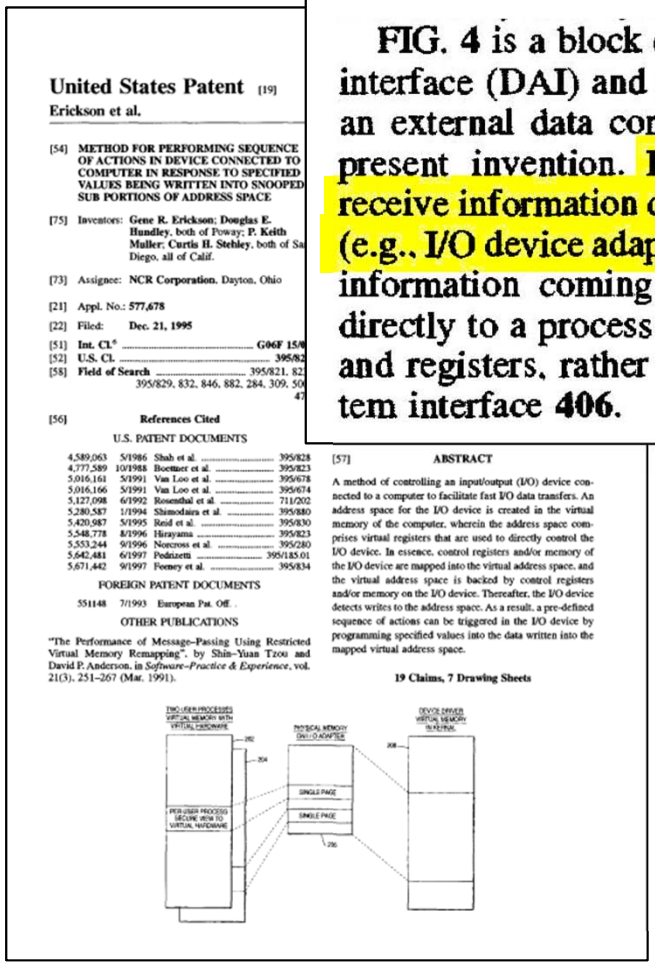
A method of controll
nected to a computer
address space for the
memory of the comp
prises virtual register
I/O device. In essenc
the I/O device are mag
the virtual address
and/or memory on the
detects writes to the a
sequence of actions
programming specific
mapped virtual addre

19 Cla

FIG. **5** is a block diagram illustrating the system organization between a main memory and an I/O device adapter memory which is compatible with the present invention. The main memory **502** implementation includes a hardware register **504** and a buffer pool **506**. The I/O device adapter implementation includes a software register **508** and a physical address buffer map **510** in the adapter's memory **512**. An endpoint table **514** in the memory **512** is used to organize multiple memory pages for individual user processes. Each entry within the endpoint table **514** points to various protocol data **518** in the memory **512** in order to accommodate multiple communication protocols, as well as previously defined protocol scripts **516** in the memory **512**, which indicate how data or information is to be transferred from the memory **512** of the I/O device adapter to the portions of main memory **502** associated with a user process.

Paper 36 (036 Opp. to Motion to Amend) at 17;
Ex. 1005 (Erickson) at 5:52-67.

Erickson discloses fast receive and transmit



United States Patent [19]

Erickson et al.

[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 Sa 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 15/0

[52] U.S. Cl. 395/82

[58] Field of Search 395/821, 82 395/829, 832, 846, 882, 284, 309, 50 47

[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).

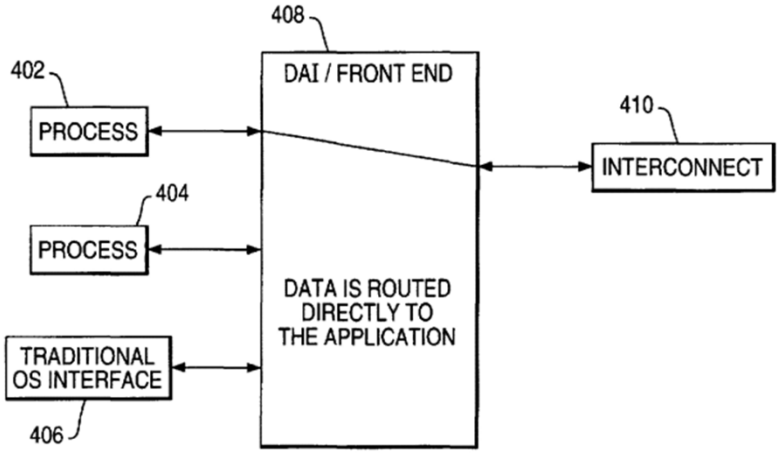
[57] ABSTRACT

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. 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**.

FIG. 4



Paper 36 (036 Opp. to Motion to Amend) at 17;
Ex. 1005 (Erickson) at 5:6-14, Fig. 4.

036 Patent: Disputes

3. Motion to Amend 036 Patent should be denied

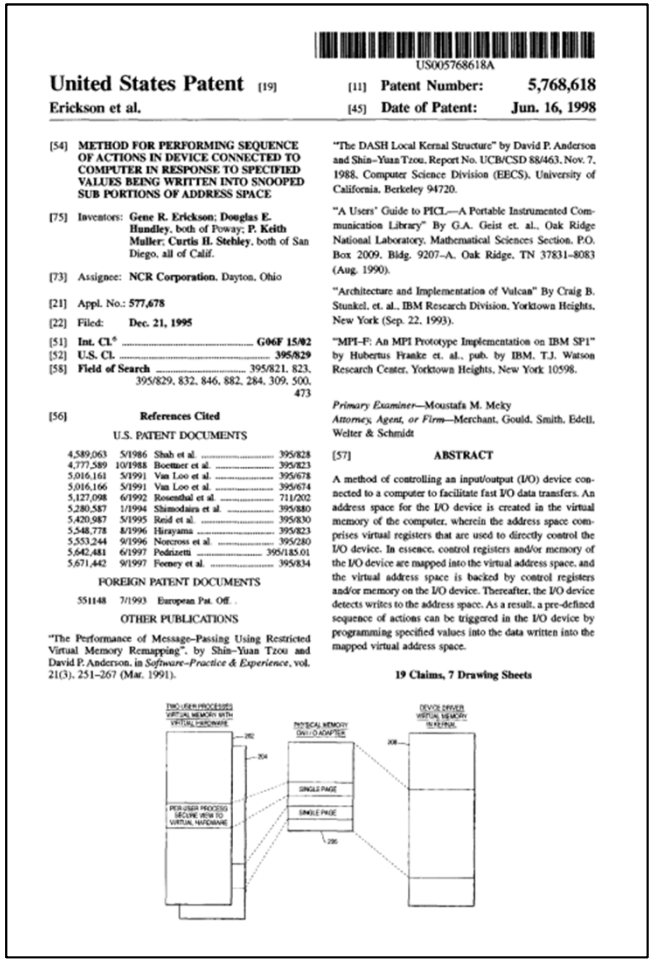
c) Substitute claims are obvious

i. Prior art discloses "the second processor determining whether an incoming message packet should be processed by second processor" (limitation 23.3)

ii. Prior art discloses "processing the incoming message packet [by the second processor]" (limitation 23.4)

iii. <u>Prior art discloses "passing the incoming message packet to the first processor for further processing" (limitation 23.5)</u>

Erickson discloses use of fast and slow applications



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).

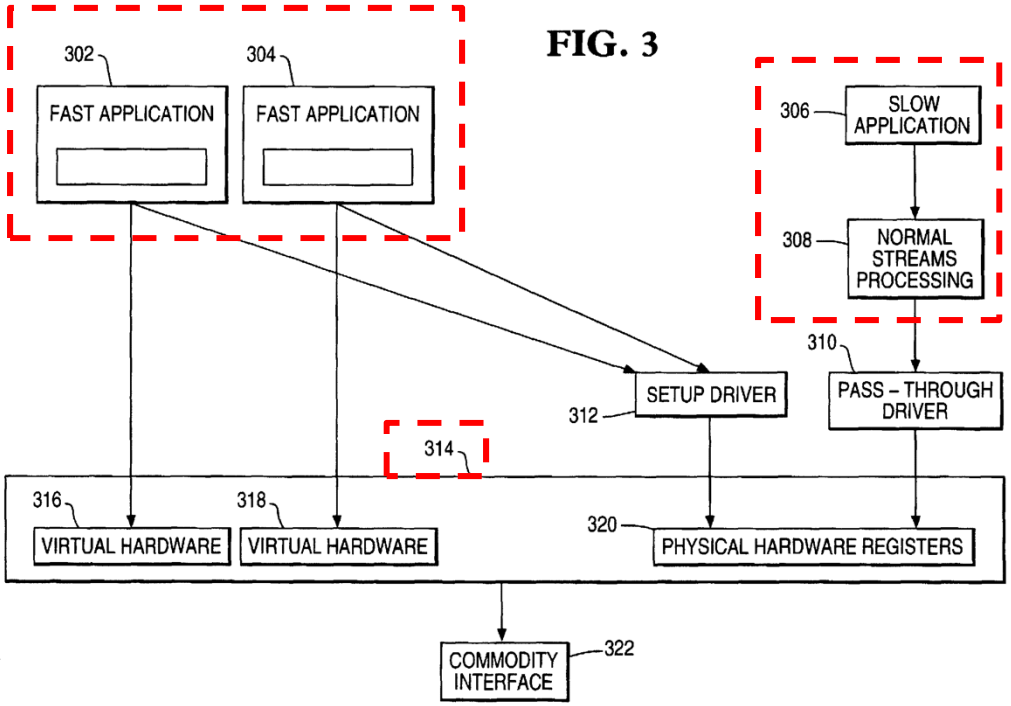
"MPI-F: An MPI Prototype Implementation on IBM SP1" by Hubertus Franke et. al., pub. by IBM, T.J. Watson Research Center, Yorktown Heights, New York 10598.

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

[57] **ABSTRACT**

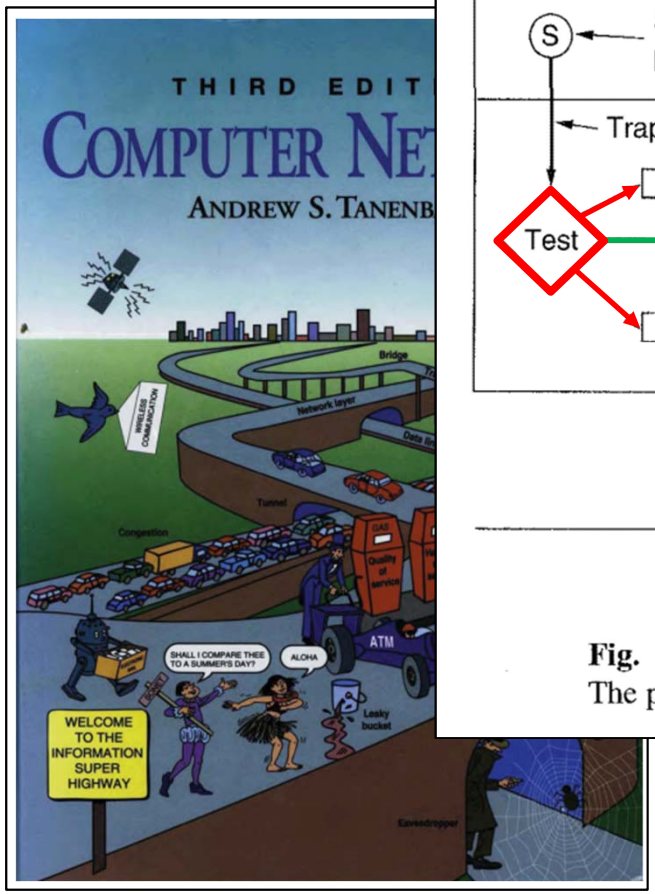
A method of controlling an input/output (I/O) device connected to a computer to facilitate fast I/O data transfers. An address space for the I/O device is created in the virtual memory of the computer, wherein the address space comprises virtual registers that are used to directly control the I/O device. 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



Paper 36 (036 Opp. to Motion to Amend) at 20;
Ex. 1003.065 (036 Horst Decl.); Ex. 1005 (Erickson) at Fig. 3.

Slow path used after determination



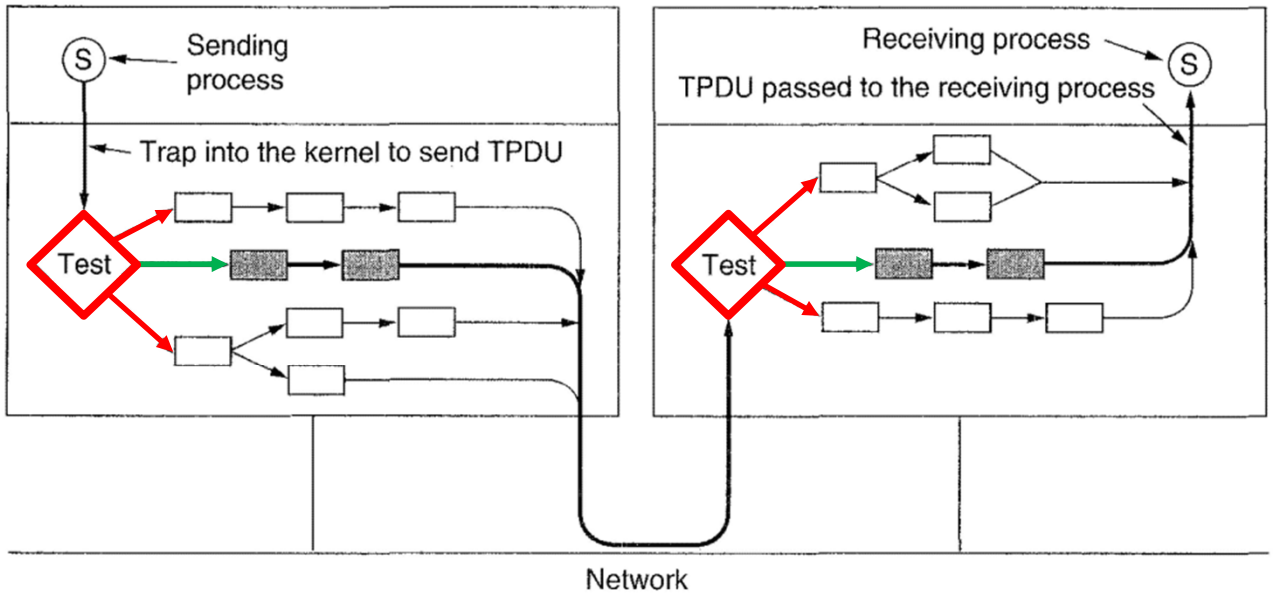


Fig. 6-49. The fast path from sender to receiver is shown with a heavy line. The processing steps on this path are shaded.

Paper 36 (036 Opp. to Motion to Amend) at 21-22;
Ex. 1006.583 -.584 (Tanenbaum96), Fig. 6-49.